UNIVERSITY OF AGDER

Parents Controlled Home Internet Cafe

Prototype of Openwrt based parental control system and web user interface design in home use router

Yunpeng Han

Supervisor
Frank Reichert

University of Agder, 2012
Faculty of Engineering and Science
Department of Information and Communication Technology

Abstract

In this report, we will present our design, implementation and testing of a router based parental control system named as ’Home Internet cafe’. The prototype is based on the OpenWrt firmware. We research the state of the art of parental control system to get a solution fit our requirements most. A design life cycle is provided for user interface design of our system from paper prototype design to web page interface mock-up prototype design, then interface building during system implementation using Html. The key functions of our system are programmed in Javascript, C and Shell script. Our implemented system provide new features include home user identification system, total time limitation functionality and user friendly interfaces. With the performance evaluation we find that the booting and processing time is harmed by our more complex algorithm, while our system can still provide normal Internet speed to our uses.

Preface

This report is my master thesis for the conclusion of my Master project on the ”Parents Controlled Home Internet Cafe” topic. I obtained a rich and varied experience during this master program. During the project work, there are successes and failures, passion and depression, ideas and problems. All these different occurrences made me gain new skills, learn new knowledge and find new possibilities in myself. And now they have become my best memories. Also I really appreciate the many people who helped me at the project.

I would firstly thank my supervisor, Frank Reichert. He gave me smart advice and new viewpoints based on my thesis work. At the same time, he taught me lots of things related to the research methods and principles of project work. I got lots of new research and work skills from him which are helpful to my thesis work and also to my future research works.

I also want to extend my gratitude to people working in RedRock AS company, Christoffer Jorgenvag, Eirik Vika and other kind men. They gave me many useful suggestions and provided user testing for my designed interface and implemented system. Also they provided me with a very friendly, comfortable workplace in their company with nice atmosphere of programming.

Finally I would thank all other people who helped me during my master thesis. Without all your supports, I won’t finish my implementation work and report as good as this presented one.

Grimstad, May 27, 2012

Yunpeng Han

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Electronic products have become an important part of people's daily life since the early 2010s. We are able to access the Internet whenever our devices can receive the wireless signals, both outside and at home. In home use cases, family members may have several devices to access the Internet: smartphones, tablet personal computers, laptops and desktop computers. The most common solution for family members to share Internet access with the only Internet interface at home is building up a home local area network with the help of a router. With most of home Internet devices work in wireless model, wireless routers are widely used to perform functionalities as wireless access points and network switches. Family members' devices and the router form a centralized server-based service model, as shown in Figure 1.1. It means that the one who control the router as the centralized server can handle the Internet use of the all Internet devices as clients. And this Internet access control is considerably in demand of family users like parents.

1.1.1 Concept of home Internet cafe

Internet cafe [23] is a common concept in modern daily life well known as a place which provides internet access to the public, usually for a fee. As the fee for using a computer is usually charged as a time-based rate, the management of timing and access control is very important in an Internet cafe. Computers in Internet cafe serving customers are client machines connected to server in the cafe and share the Internet access from the server. Local area network model in Internet cafe is similar to the wireless local area network in a family using router to combine different devices. So we can build up a home Internet cafe with the access control management consulting the working model of public Internet cafe, of course without the fee for using Internet.

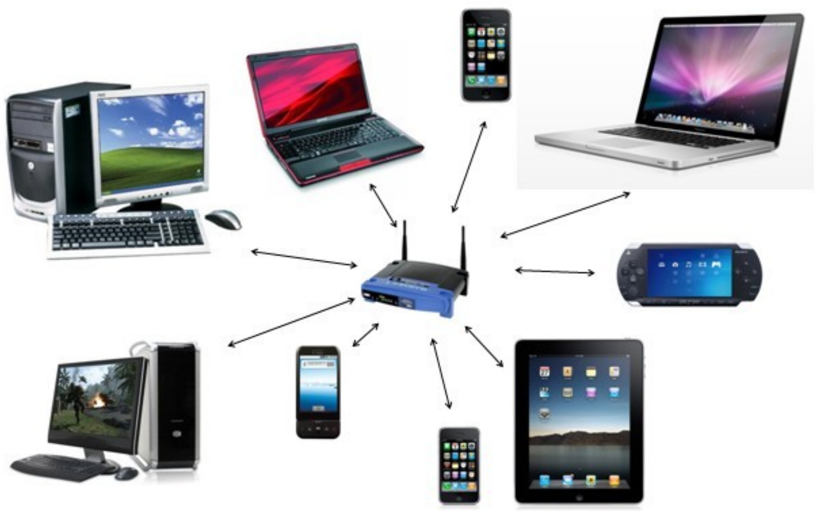


Figure 1.1: Home local area network with a router

1.2 Motivation

We have mentioned that family members can easily access Internet with their own devices through the router. Nowadays many children in the family own electronics products. As children like to surf Internet at late night and play on-line games, parents often worry about their health getting harmed by staying up till midnight, watching the screen for too long and staying in the room without exercise outdoors. Therefore parents require a method to control family members' Internet usage. All routers sold have access control functionalities. But it's hard for most parents to understand how to control their children's Internet access via IP or MAC addresses in cases they lack the necessary IT skills. There are some third party parent control applications friendly to Internet illiterates. But many of them are charged for fees and need to be implemented in children's devices. Considering these limitations, a free and user friendly Internet access control system will be very helpful to parents. So we decide to build up a system like this. Since the system works similar to a Internet cafe at home, we name it as 'Home Internet cafe'.

1.2.1 Parents' Demand of Functionalities

Since our task is to set up home Internet cafe functions that help one to easily manage the family members' use of the Internet, mainly considering the demand of parents to control their children's use of the Internet, the management should contain:

- Set up the days that an appointed user is allowed to access the Internet
- Set up the time in a day when an appointed user is allowed to access the Internet

- Set up the total Internet access limitation time of target user
- Block some services like catch P2P protocols or block access to Xboxlive
- Website blocking by URL Address or Keyword

Let's suppose a scenario for access control in a family with two children. Children have to go to school during workdays so parents hope they can go to bed early at night. Therefore to stop the children from playing online games or surfing the Internet late, their Internet access will be shut down after 21:00 and activated next afternoon when children arrive home. During weekends and holidays, another rule will be used which blocks children's use of Internet between 00:00 am and 6:00 to avoid them stay up late till midnight.

1.2.2 Limitations of Ordinary Routers

As devices in family all connect to router and share the Internet access from the router, it is simple and efficient to implement our home Internet cafe controller inside the router. Routers sold in the market usually have their own control panels developed by the companies. But these control panels only contain basic access restrictions like block/allow Internet access by media access control (MAC) address without functions we expected. To improve the services and functionalities of routers to support our implement of home Internet cafe, we can ask for help from third party developed firmware for home used routers to add features for access control and other more functions.

1.2.3 Unfriendly User Interface

The user interfaces of normal routers are usually provided as web interfaces where users can login via username and password and configure many different user settings. And many configuration options are not friendly for users lacking of Internet knowledge. For example, many systems require users to type target mac address or ip address ranges when set up rules for time restrictions. In our home Internet cafe system we plan to provide a graphical user interface (GUI) with basic settings typed and advanced functions operated via simple buttons and checkboxes instead of choosing or typing different complex settings requiring Internet skills. Then we can make it possible for computer beginners to operate easily by clicking their mouses to handle our home Internet cafe system.

1.3 Goal and Approach

Our main goal in this thesis is to design a user friendly Internet access control system for family users, named as home Internet cafe. We will compare different solutions with existing access control technologies to figure out which is the best and design our system based on the chosen technology.

After that we will design our user Interface via paper prototype and mockup tools and have the interface tested by different users to improve the usability and make it more user-friendly. Then we will turn the proposed solution and designed interface into an application system with a web interface. During this task we will write our own code to contribute more access control functionalities for home users under the opensource firmware. After the system is built, we will have it tested by different users and try all developed functionalities to prove that the system fit the requirement and then measure its performance.

1.4 Plan of Development Process

Our main task in this thesis is to develop a home Internet cafe system with a graphical user interface for users to operate the routers and set the parameters for different Internet access control functionalities. We can separate this task into the following steps:

1. Start our work with viewing the state of the art of different parent control systems to find out the importance of different parents requirements and choose the solution that fit our design issues most.

2. Design the system structure based on the solutions we found and figure out the existing features and the features to add. Design related functions and algorithm to realize the new features.

3. Design the user interface in two steps from paper prototype to web interface mockup. In each step we invite several users to test the interface usability, get the feedback and improve the interface.

4. Read the code of the opensoure software router firmware and list out the access control functions in the software and the functions required but not realized in the software. Then add new modules based on the current firmware to implement new access control functions for home Internet cafe.

5. Add new configurations of new functions developed in the target firmware's control panel through web browser. Check how are the commands and data sent and received between devices and routers and verify the interfaces used during the configuration. This is a preparation for developing a new graphical user interface as an application in new operating system.

6. Start the programming using several programming languages with the different application development tools to build an application as a GUI with the template user interface designed to operate the routers. The user interface built should be user-friendly to ordinary people with less IT knowledge.

7. Implement the developed system as a firmware for routers. Have the systems web interface and the access control functions validated in the real use cases and check their performance.

We combine the system design and implementation work and report writing work to make a time table for our thesis shows in Figure 1.2.

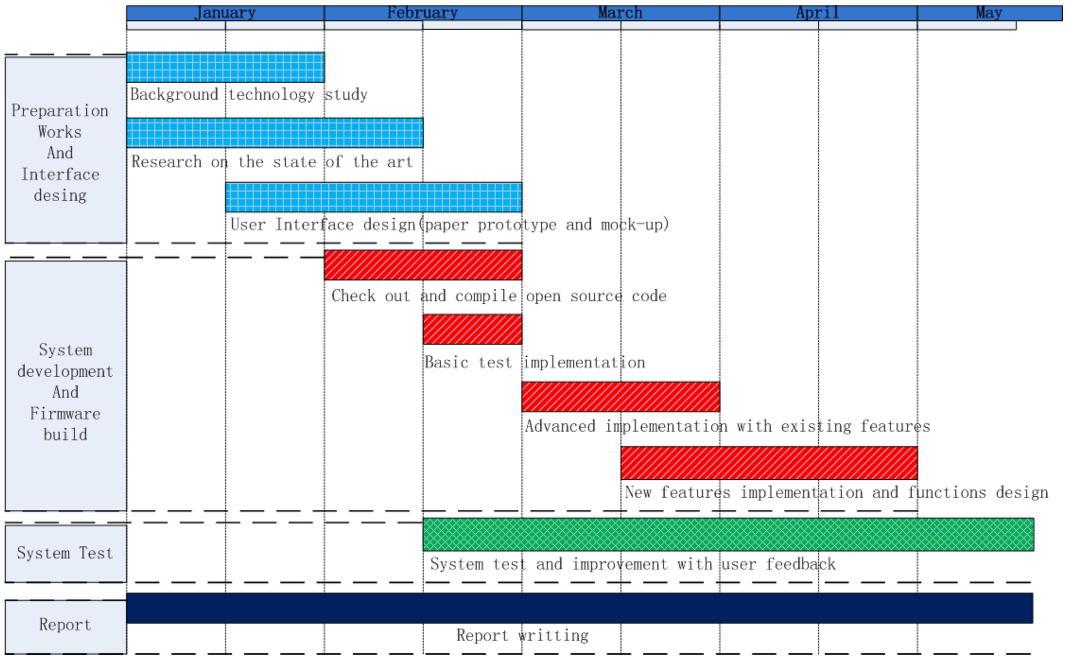


Figure 1.2: The timetable of thesis

1.5 Method

Different methods are used in each step of our thesis work. In the user interface development part, we first use paper prototype to design the interface and then turn the improved interface into web pages for operator to test via mockup tool Pencil Project [22].

Our goal is to implement a new system in home user router to provide different functions for parent users. The router we choose for implementation is a Linksys Wireless-G broadband router WRT54GL v1.1 [6]. And we use OpenWrt [19] as the basic opensource firmware to develop functions and add features. The OpenWrt firmware code is checked out and compiled under Ubuntu 11.04 linux operating system. Then we design functions and build user interface using several programming languages including C, Javascript, html and shell script. During the development we use debug tools like online syntax check tool 'Javascript lint' [25] and web page debug tool 'Firebug' in the Firefox browser [17]. We combine these tools to reach our goal of a home Internet cafe system design and implementation.

1.6 Report outline

The rest of the report is organized as follow:

In Chapter 2, we introduce the state of the art in parental control system with different solutions implemented in different family users' devices. We make comparison among different solutions to find advantages and disadvantages of them.

In Chapter 3, we make argument about why we choose the router based system as our solution. Then we discuss our design issues, especially our user interface design method. Also we give out the implementation plan.

In Chapter 4, we show our implementation works in details step by step from basic implementation to key functions design. And the debug tools are also introduced since debugging is an important part of implementation.

In Chapter 5, we provide the overview of user interface containing most of the key functions. Some testing data is present in this chapter as performance evaluation results.

In Chapter 6, we make conclusion about our final system. The advantages are discussed to show our contribution. And the disadvantages are discussed to give prospects for further works.

Chapter 2

State of the Art

In this chapter we will introduce different solutions for parent Internet control including normal routers control panels, related third party router firmwares and some commercial parental control products. The introduction and comparison will state the advantages and disadvantages to help us come up with a solution fit the requirement most.

2.1 Access control features

Our task is to set up home Internet cafe system that help one to easily manage the family members' use of the Internet. Parental controls contain main features like content filters, usage controls and monitoring. Mainly considering the demand of parents to control their children's usage of the Internet, the management in our system should contain:

- Confirm and control users' Internet access directly
- Set up the time when an appointed user is allowed to access the Internet
- Configure time limits for Internet usage
- Block selected Internet services
- Web sites blocking

These functionalities are realized by different parental control systems in different ways. But some systems mainly focus on one functionality. Some systems are well built with all functions but they charge for fee. We will introduce these different systems in the following section.

2.2 Parental control solutions

To realize the web sites blocking and usage management, it is common to use filters for web sites and services blocking and use timepieces for time limitations. We will introduce different applications and systems grouped by where their filters and timepieces located.

2.2.1 Control systems in devices of target users

Many parental control systems are implemented in target users' devices. These systems ranges from web browsers, programs to operating systems itself.

We know operating systems like Mac OS X, Windows 7 and some Linux versions all have built-in parental control features. With different authority between administrator account and normal accounts in OS, it is convenient in cases that parent and children share the same computer.

Also web browsers have parental control plugins, like Glubble[24] for Firefox. And there are web browsers specially developed for children like Buddy Browser[5], KidZui[16]. Also in mobile devices like iPhone and iPad, we have special browser like SurfBalance[1]. These browsers can provide web site filters and time limitation functions during Internet surfing, but only if they are used as the surf tools.

2.2.2 Control systems in device of the handler

Many third party programs are developed to provide parental control installed at user side, but most of them have a server side in parents' devices. Famous applications like Safe Eyes[15], Norton Online family[27] and Net Nanny[7] all work in server and clients mode in which parents' devices act as server side and give out license for client devices with content analyzed and blocked in real time. And parents can get report of the actions done by client users if usage monitoring functions are contained. Mobile devices have similar applications like Watchdog[10] and safeeyes mobile. Note that some parental control systems give out notification for client users telling them the Internet usage is limited. And some systems do it in stealth mode, mainly when monitoring functions are active. Figure 2.1 shows the control panel of safeeyes.

2.2.3 Control systems in router

As we mentioned, routers are widely used in homes. With certain IT skills, parents can customize Internet usage by deploying self-configured routers.

Basic router functions

Most routers selling in the market provide functionality of access restrictions. The figure 2.2

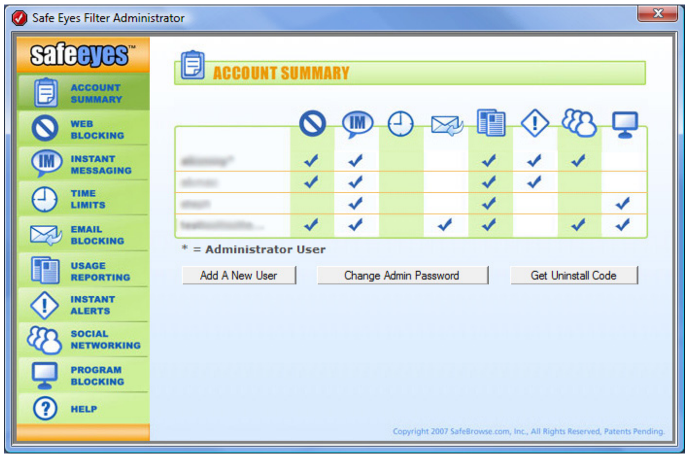
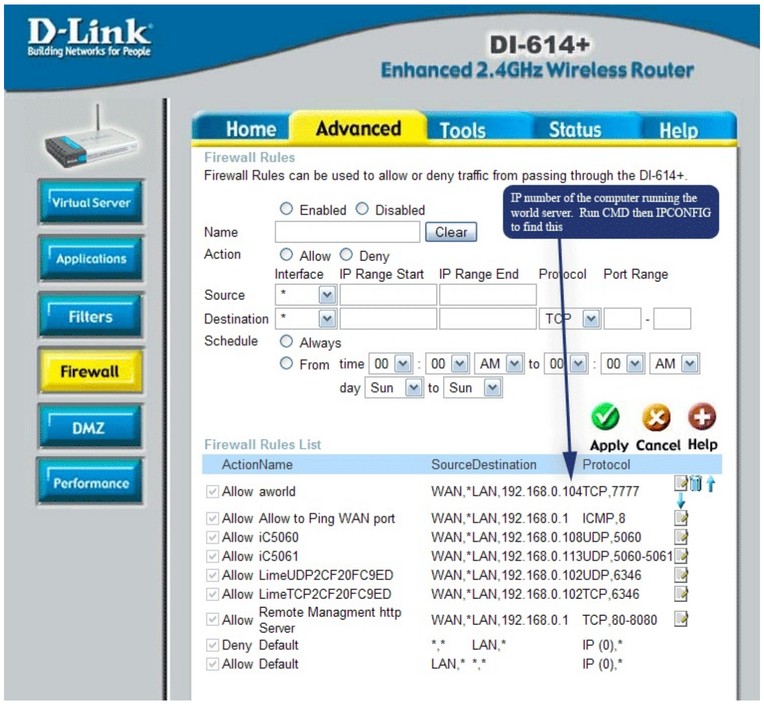


Figure 2.1: The control panel of Safe eyes [15]

shows a router control panel of D-Link DI-614+ Wireless Router. The firewall rules used to allow or deny traffic passing through the router can be used for usage management for appointed user. But it only works when every user in the WLAN has static IP address based on MAC address. This condition need more configuration and some IT skills.


D-Link
Building Networks for People
DI-614+
Enhanced 2.4GHz Wireless Router
Home
Advanced
Tools
Status
Help
Virtual Server
Applications
Filters
Firewall
DMZ
Performance
IP number of the computer running the world server. Run CMD then IPCONFIG to find this
Clear
Apply
Cancel
Help

Firewall Rules
Firewall Rules can be used to allow or deny traffic from passing through the DI-614+.

	○ Enabled ○ Disabled
Name	
Action	○ Allow ○ Deny
	Interface / IP Range Start / IP Range End / Protocol / Port Range
Source	*
Destination	* / TCP
Schedule	○ Always
	○ From time 00 : 00 AM to 00 : 00 AM
	day Sun to Sun

Firewall Rules List

Action	Name	Source	Destination	Protocol
Allow	aworld	WAN,*	LAN,192.168.0.104	TCP,7777
Allow	Allow to Ping WAN port	WAN,*	LAN,192.168.0.1	ICMP,8
Allow	iC5060	WAN,*	LAN,192.168.0.108	UDP,5060
Allow	iC5061	WAN,*	LAN,192.168.0.113	UDP,5060-5061
Allow	LimeUDP2CF20FC9ED	WAN,*	LAN,192.168.0.102	UDP,6346
Allow	LimeTCP2CF20FC9ED	WAN,*	LAN,192.168.0.102	TCP,6346
Allow	Remote Managment http Server	WAN,*	LAN,192.168.0.1	TCP,80-8080
Deny	Default	*,*	LAN,*	IP (0),*
Allow	Default	LAN,*	*,*	IP (0),*

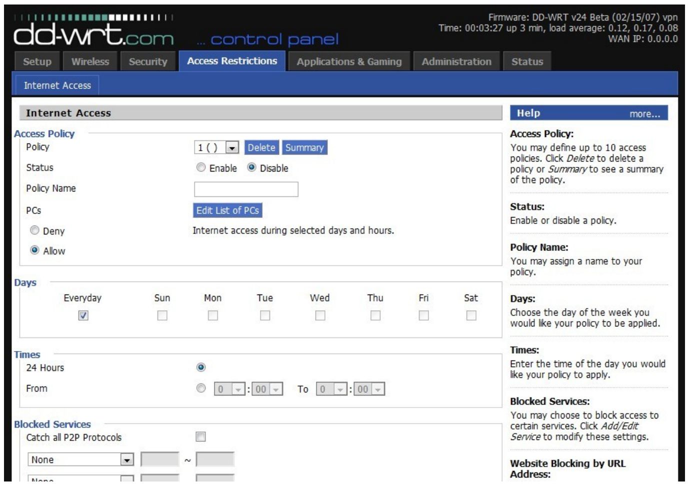
Figure 2.2: The control panel of D-Link router [6]

Advanced third party firmwares

There are several third party firmwares for routers developed by groups other than the manufacturer of those products. And these firmwares extend new functionalities for home use routers and make it easy to use for ordinary people with new friendly control panels. Famous third part firmwares are OpenWrt [19] , DD-WRT[9], DebWRT and others. We will introduce DD-WRT as it is well developed and is provided as an open source project.

DD-WRT is a third party developed firmware released under the terms of the GPL for many ieee802.11a/b/g/h/n wireless routers based on a Broadcom or Atheros chip reference design. The firmware is maintained by BrainSlayer and is hosted at dd-wrt.com. The first versions of DD-WRT were based on the Alchemy Firmware from Sveasoft Inc, which is in turn based on the original GPL'd Linksys firmware and a number of other open source projects [9].

DD-WRT offers many advanced features not found in the OEM firmwares of those routers. Figure 2.3 shows the control panel of DD-WRT in the access restriction part. Users can directly set rules for appointed users by operating with Mac addresses and rules are easily added on the panel. However users still need basic IT skills.



Firmware: DD-WRT v24 Beta (02/15/07) vpn
Time: 00:03:27 up 3 min, load average: 0.12, 0.17, 0.08
WAN IP: 0.0.0.0

Internet Access

Access Policy
Policy: 1 ( ) Delete Summary
Status: Enable [✓ Disable]
Policy Name:
PCs: Edit List of PCs
Deny
[✓ Allow] Internet access during selected days and hours.

Days
[✓ Everyday] Sun Mon Tue Wed Thu Fri Sat

Times
[✓ 24 Hours]
From 0 : 00 To 0 : 00

Blocked Services
Catch all P2P Protocols
None ~

Access Policy:
You may define up to 10 access policies. Click *Delete* to delete a policy or *Summary* to see a summary of the policy.

Status:
Enable or disable a policy.

Policy Name:
You may assign a name to your policy.

Days:
Choose the day of the week you would like your policy to be applied.

Times:
Enter the time of the day you would like your policy to apply.

Blocked Services:
You may choose to block access to certain services. Click *Add/Edit Service* to modify these settings.

Website Blocking by URL Address:

Figure 2.3: The control panel of DD-WRT [8]

2.3 Comparison and Discussion of different Solutions

Different solutions exist aimed at parental control with web sites blocking and usage controls as we introduced above. They have different principles and appear as different applications located in different devices. To propose our own solution for home Internet cafe, we compare these different solutions for later reference.

For the parental control functions implemented in the operating systems, they mostly work in cases that parent and children share the same computer. Nowadays family members all have their own devices. Children will revolt against the one who set an administrator account in their devices to restrict their usage. And it is easy to avoid the control by re-installing the operating system or using another operating system. The situation is the same for specially designed web browsers. To bypass these application we just need to use a different browser. So these two solutions only fit the requirement for parents who have very young kids and want to protect them from unintentional wrong Internet usage.

Those third-party systems specially designed for parental control have much better performances. No matter working in server-client model or implemented in target users' devices, These control systems are hard to bypass as some special features are designed against bypassing. The same way to bypass these systems is choose an alternative operating system without them. Also proxy servers are used if website block is realized by URL blocking. For these solutions, some people may feel it annoying to see the notification of the parental control system like "you are currently under the watching of safeeyes!". While they also will get angry if they find themselves monitored by stealthy programs.

The systems implemented in target devices are user-friendly for users lack of IT skills to configure and operate. That is because they are specially designed to fit parent users' requirement in Internet usage management. While for access control solutions realized by routers, they are always designed as additional functions which have to give way to the main process of routers as forwarding data packets. As advanced settings, most router control panels are not so user-friendly to Internet illiterates in the access restriction configuration part. And most access restrictions in routers provide less functions which can't fulfill different Internet usage management requirements.

On the other hand, access restrictions realized in the router side do not effect by target user's system environment. It means that people have less ways to bypass the access control in the router side with limited authority. Also as the routers handle all devices' Internet access, the configuration of access restriction is more convenient and there is no need to install and update software in all home devices.

Chapter 3

Proposed Solution

We have presented our requirements and discussed the existing solutions with advantages and disadvantages above. In this chapter we will propose our solution and explain the design in details.

3.1 Proposed solution

Our goal is to design a parent controlled home Internet cafe system. As we have compared different kind of existing parent controlled Internet usage management system, we can find that to propose a router based system is the best choice. The router based system will have the following advantages:

- As a true router-based solution, nothing is needed to be installed on users' computers.
- The system can control all computers in home network.
- User can set up time limit of Internet use for different home members with different rules separately and conveniently.
- The system is OS independent that works on both Windows, Linux and Mac OS.
- This is a free user system. We don't need to pay for it.

Our router based system is shown in Figure 3.1. Data packets sent from Internet or home users are managed by the filter in the router based on the pre-set rules. If the packet is sent from/to a user blocked at current time or using a blocked protocol , it will be dropped. Otherwise it will be forwarded to its target through the router.

What's the key functionalities for parent controlled home internet cafe? We should always remember to focus on the Internet user management part. In our design, the system has a parent account, which takes the role of normal administrator. The parent account can set router features,

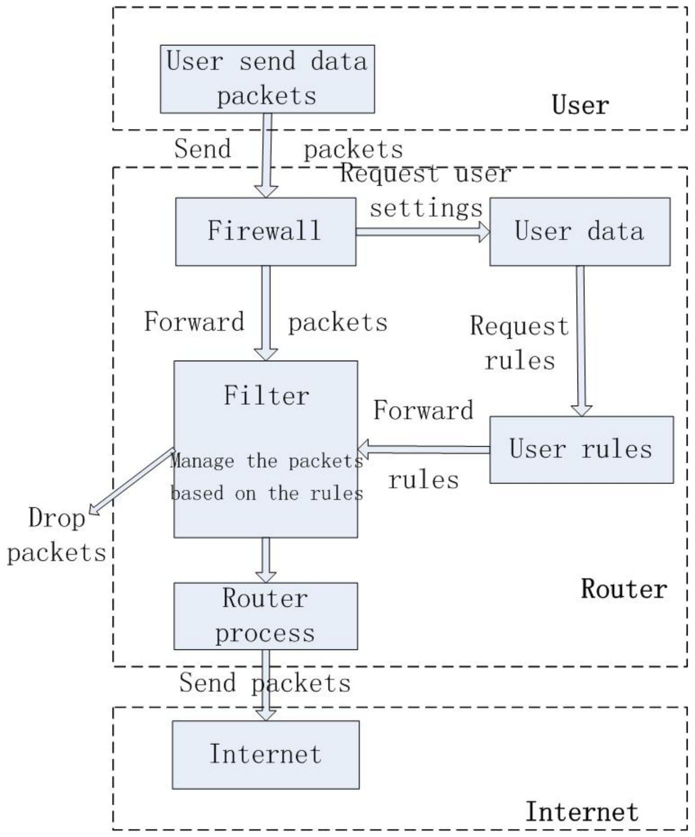


Figure 3.1: Router based system structure

change different settings and the most important: handle the Internet usage of all hosts connected to the router. We introduce new features such as Internet code and family user. Internet code and family user are introduced to make parent account easier to manage the whole Internet. Family member use an Internet code given by parent account to access the Internet through the router, which make their devices attached to the user data stored in the system. By setting rules based on the Internet code related to home member, parent account can easily manage the Internet use of different home members regardless of what devices target users are using in case that they have several devices.

3.2 Discussion of design issues

Although we decide to develop a router based system for home Internet use control, it is totally impossible to build a whole new firmware with drivers, linux kernel and user interfaces all built by ourselves. Instead, we will work on improving existing firmware, designing a user friendly interface for it and adding new management features that parents need.

Normal router firmwares provided by company usually have user interface that is easy to use but with less advanced features. In the other hand, there are many third party firmwares providing interesting and useful functionalities, but their interfaces are usually complex. In our case for the implementation, as third party firmwares can provide more features we need and since they are open source which can be easier to access and modify, we decide to design our solution based on a third party firmware.

At first we consider DD-WRT [9] as introduced in section 2.2.3. It is a really good firmware providing many advanced functionalities. However, the source code of the user interface part in DD-WRT is protected from full edit to avoid people from re-banding the firmware to sell it. As one of our purpose is to provide user friendly user interface for people lack of IT skills, We need to design new user interface. It makes DD-WRT not available for implementation.

Later we go to OpenWRT [19], whose web interface is fully editable. As a router firmware OpenWRT has less features than DD-WRT but those features are enough to implement our solution for normal family use. OpenWRT is a Linux distribution for router and works with thousands of features based on different software packages freely chosen by users. The web interface is also realized via software packages and we have 3 different choice:

- **LuCI:** written in Lua(programming language)/Shell(for linux)
- **X-WRT:** written in HTML(programming language)/Shell(for linux)
- **Gargoyle:** written in Javascript(programming language)/Shell(for linux)

As both the three packages can provide user interface, we pick Gargoyle [11] as the base structure for our implementation as we are familiar with Javascript.

3.3 User interface design

After decide the basic firmware to use, the next task is the user interface design. As our target is not only to provide a system with advanced features fit for family use but also user friendly interface for normal users. It is very important to provide a well designed user interface to satisfy users and facilitate easy usage.

3.3.1 Principle of interface design

Before we start our work of user interface design, it is very important to understand what features should a real user-friendly interface provide to users. Steve Krug wrote in his book ”Don’t make me think: A common sense approach to web usability” [26] that web interface designer should provide user a interface that users can ”get it” -What it is and how to user it- without expending any effort thinking about it. It means that our goal is to make each interface page to be self-evident, so that the average user will know what it is and how to use it by looking at it. Based on this key issue, we should work on many principles when design a user-friendly interface:

Think as a user when design

We always want to design interface with much details and give out all information user may need to help them operate the system. We might create lots of buttons with descriptions and different enable boxes or provide other choose methods and hope they can cover all user needs and make it easy for the user to browse. But the fact in life is that users always operate the interface in a different way as we thought. Most users don't have the patience and know that they don't have to read everything in the interface page. It makes descriptions not that helpful. Also users make out choice directly when they think it will be helpful. The choice may not be the optimal one as users won't view all available options and pick out the best one. It makes lots of options cover all user needs confusing to users in some cases. Also users always muddle through when they can't figure out how things work. They don't care how target function works and may always do things beyond designers' intention. All these user behaviors are common during interface usage and should be considered during interface design to avoid useless or confusing features for users added. When designing a interface, we should think as a user to fit to his way of operating the interface instead of as a designer intending the user to do specific operations.

Provide a clear visual hierarchy

When think as a user we know that user always go through interface fast just to find things they interest in, so we should provide a interface page with all of the visual options and features displayed clearly with their relationships accurately presented. Users should then be able to pick out options they need fast and understand which things are related and which things are contained in other things. That is, we should provide a clear visual hierarchy. Marking more important options with more prominent fonts and colors, showing logically related features with related visual styles and combining things as nested to show what's part of what are all useful principles for this work.

Divide interface into clearly defined areas

We save users' time by dividing page into clearly defined areas to allow them to decide which areas to focus on and which areas they can ignore. With prominent area information users can decide very quickly which parts of the page are likely to have useful information for them and also avoid confusions.

Make operate options obvious

Users always look for the next thing to click or choose when they find things interested. It's very important to make it obvious what is clickable. When provide functions or links to users, it is better to provide via clearly marked clickable buttons, checkboxes or other symbols than to use clickable highlight or colorful text or pictures with some notifications inviting users to click.

3.3.2 Interface design life cycle

After we get the basic idea of Home Internet cafe, we need to design our user interface. We only know that parents need a user friendly interface to operate the system to manage their family members' Internet use. To gain user expectations for understanding the necessary of different functionalities and to get user experience and user acceptance for the to be built user interface, we need a more practical design life cycle instead of describing how the interface looks like by language and then directly going to programming part. The design life cycle is shown in Figure 3.2.

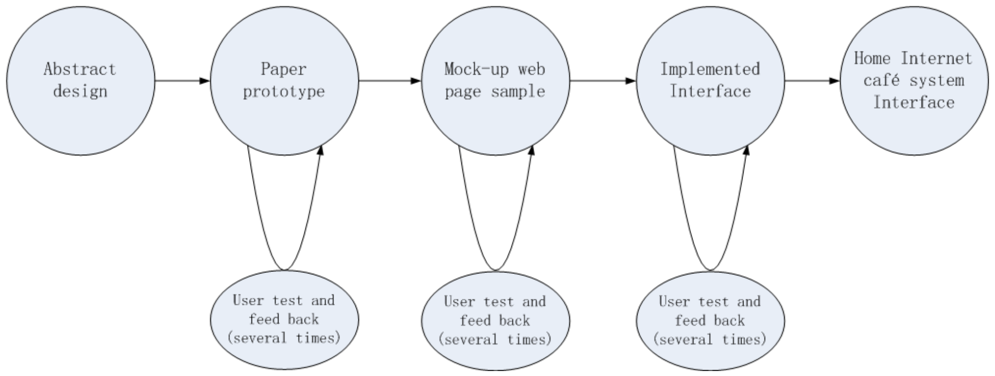
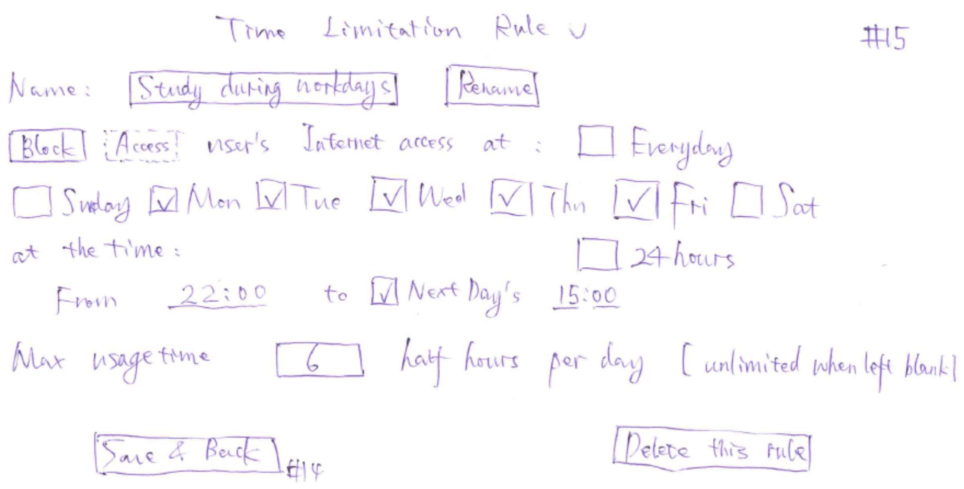

Abstract design
Paper prototype
Mock-up web page sample
Implemented Interface
Home Internet café system Interface
User test and feed back (several times)
User test and feed back (several times)
User test and feed back (several times)

Figure 3.2: User interface design life cycle

To provide a practical interface design life cycle, we introduce the paper prototype to start our design. To make paper prototype, we give out many pieces of paper, each as a web page with functions and buttons on it. By virtually 'click' a button on the paper web page we give out another related paper page, just like serving a web interface abstractly. One of the paper prototype page is shown in Figure 3.3. We pick several users and let them go through the paper pages and record those features that make our user confused or stuck. Since the prototype is written on paper, we can easily change the prototype based on users' advices instead of writing lots of code again and again. By considering not only the usability, which considers whether the system is easy to use, but also the usefulness, which considers whether the system features are important and necessary for home users, we manage to get a draft paper prototype as the basic idea of our user interface and functionality. The paper prototype is shown in the Appendix C.

Then we go to a step named as "mock-up". Mock-up is a prototype provides at least part of the functionality of our system. It is the advanced step to continue acquiring feedback from users with real software interface pages instead of paper. But although given as web pages, clicking some buttons won't really provide the related functions. In this way we can continue to verify the user experience and fix the interface without big changes in code. The mock-up interface is made by a mock-up tool named Pencil Project [22] as an add-on of the Firefox browser. One page of the prototype is shown in Figure 3.4 and all the mock pages are attached in the Appendix D.



Time Limitation Rule ∨ #15

Name: Study during workdays Rename

Block Access user's Internet access at: ☐ Everyday

☐ Sunday ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☐ Sat

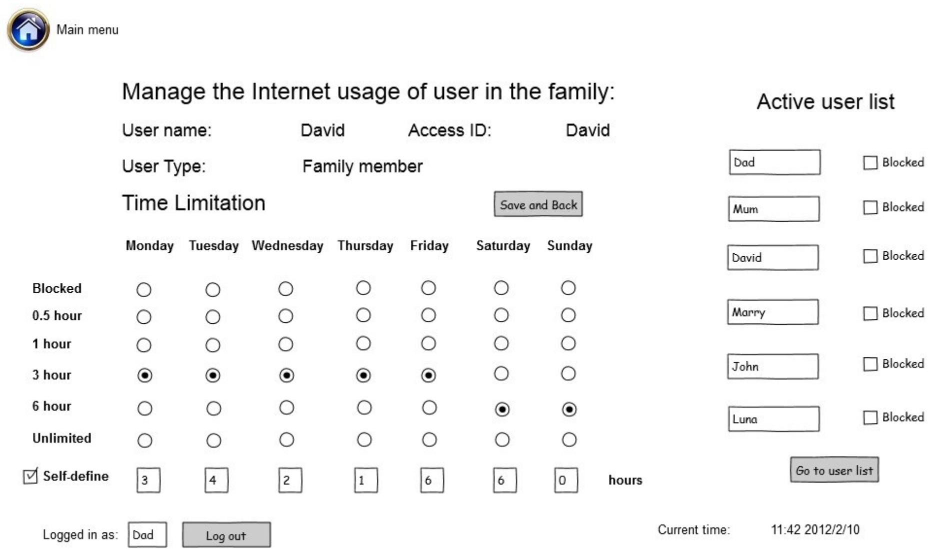
at the time: ☐ 24 hours

From 22:00 to ☑ Next Day's 15:00

Max usage time 6 half hours per day [unlimited when left blank]

Save & Back #14 Delete this rule

Figure 3.3: A sample page of the interface design paper prototype



Manage the Internet usage of user in the family:

User name: David Access ID: David

User Type: Family member

Time Limitation

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Blocked	○	○	○	○	○	○	○
0.5 hour	○	○	○	○	○	○	○
1 hour	○	○	○	○	○	○	○
3 hour	●	●	●	●	●	○	○
6 hour	○	○	○	○	○	●	●
Unlimited	○	○	○	○	○	○	○
☑ Self-define	3	4	2	1	6	6	0

hours

Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☐ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Logged in as: Dad

Current time: 11:42 2012/2/10

Figure 3.4: Mock-up user interface page

3.4 Usability testing with user feedback

During both paper prototype design and mock-up interface page design, we improve our designed interface based on the testing result of target users for many times. This work of usability testing is very important because experiments are the most effective way to find out problems and get new ideas. A user friendly interface is not built up via designer's thought that 'it should be a user friendly interface if built like this'. We should have users test the interface to verify the usability with continuous testing and improving based on the feedback. However, to get more useful feedback and to find more errors and bugs, effective way of user testing is very important. In paper prototype test, we provide different paper pages to users acting like web pages that allow them to do 'virtual click' on the buttons draw in the paper and give our feedback with another paper page. In mock-up web pages we have our users go through pages like real interface without having functions operating based on user action. In implemented interface test we are providing our system to users. Users can do different actions, get the feedback of the system and see the result of the executed functions.

During the design, we get to know our interface very well. It is in another word 'too much' in usability testing. We need to find others who know nothing about our development to only view and operate our system using the interface. The first thing for our tests are the users chosen. They can be anyone regardless of their Internet skills or whether they have been a parent. It is because we always can get new information when observing others operating the interface, no matter if they always face problems or can use it without getting confused. It is better to have a user that lack computer skills to test our system. As he may give us more ideas in improving interface at the later part of our design. But at the very beginning with many key features not built up in each step of the design life cycle, normal user will be more helpful.

Another test issue is the times of testing together with number of test users. We usually spend less than half a hour to have short test with users. To have four users test twice and to have eight users test once will cost the same time if we observe their operation separately. More users test may find more problems in a single test, but in a second test with problems found in the first time test fixed, users may find other new problems ignored the first time round. It makes the more times of testing with less test users more effective.

The test task itself will be very simple based on what we are developing. Our system is a parent control system. So our test scenario is always that users are acting as parents to control Internet usage of the children. One of the most important thing to keep in mind is that during the test anything could happen. It is not possible for us to handle what our users will do when operating. And it is very important to avoid giving out hints or stopping users from doing something unless it is really needed to continue the test. A successful system should be tolerable to gruff actions and stupid mistakes. And user need hint because they get confused, then what we should do is to record it and fix it in later design. Also it is very import to keep patient and calm down when we find that user are facing problems, some are caused by themselves and some are due to our system bugs. You never know what your users will do. During paper prototype and mock-up page test things could be easy. But in the implement interface test, we may face problems like system crashing or functions not working many times. As a example, our system crashed in a case that our test user created two home users with the same name and password - since we didn't state that user is not allowed to do something and didn't correct the mistakes, user will make unexpected actions like this. It is helpful based on this kind of 'stupid' operations as they bring us problems in the system and help us to improve the interface. Any errors users caused when their operations are not forbidden by the system should be interface problems of designer, not the user themselves. During the test observing , we should always take notes of problems found and suggestions. Our interface is improved with these problems fixed and user suggestions applied. Testing is very important to provide a user friendly system, especially in our case to develop a system for normal home users.

3.5 Implementation plan

After the work of design with paper prototype and mock-up, we can start our implementation based on the designed user interface template. Our target is to implement our prototype designed via paper and mock-up in a OpenWRT based router as a home Internet cafe system. Different features exist in the prototype and some are realized by existing OpenWRT software while some features need to be added by ourselves. For the implementation, we plan to do it step by step as following:

1. Try to add our own page like home Internet cafe welcome page into the existed firmware
2. Add interface features with functionalities existing in OpenWRT firmware:
    - Time limitation
    - Restriction exception
    - web sites and protocol filter
3. Develop the two main new functionalities:
    - Home user code identify system
    - total use time limitation
4. Combine features to complete the user interface

Chapter 4

Implementation

After design we turn our idea and solution into prototype and have it tested by different users to verify and improve the user interface and functions. The next step is to implement the prototype in router to build up a real home Internet cafe system. In this chapter we will describe the implementation of our home Internet cafe prototype in the router.

4.1 Firmware and devices

Our implementation is based on the OpenWRT [19] firmware installed in a Linksys Wireless-G broadband router WRT54GL v1.1 [6]. To start the implementation, we should get a good understanding on target firmware and devices. Figure 4.1 shows the picture of our router devices and development set up scenario.

4.1.1 OpenWRT firmware and router structure

Normal home routers are edge routers placed between ISP network and home users. They have limited memory and small CPU for basic use. In the home routers, the OpenWrt firmware act as a Linux distribution operating system based on the router hardwares that handle users' Internet connections and packets. It works with drivers to the hardware and provide different functions as software applications to home users. The time limitation functions and web user interface are software applications based on the Linux OpenWrt firmware.

As different routers have different hardware, the OpenWrt as the operating system must be built with specific drivers for target router. In our case we use the Linksys router WRT54GL v1.1, whose CPU is broadcom BCM5352@200MHz [4]. The default built firmwares are shown as image file to be uploaded specially built for broadcom chips.

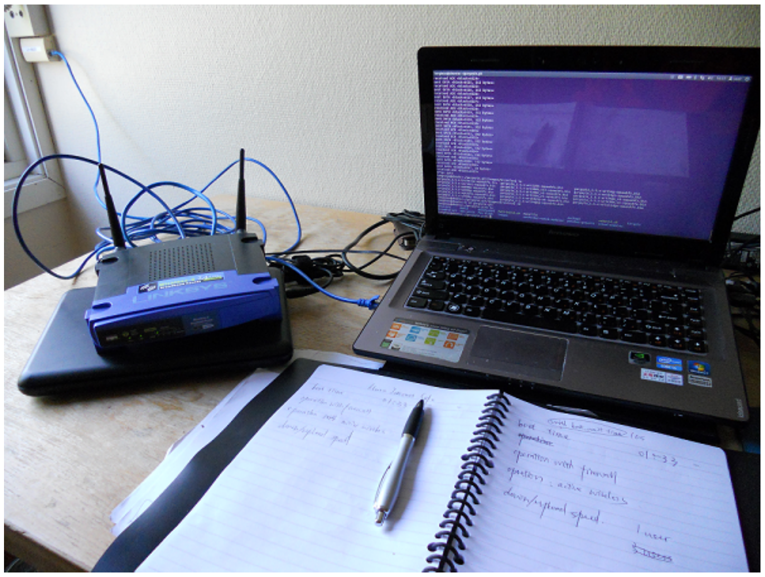


Figure 4.1: Implementation scenario

4.2 Basic Implementation with existing features

Our first step is to get the open source code of OpenWrt, compile it and build our own firmware with small features changed. Then we upload the firmware image we built and check whether the changes applied or not. We can do lots of improvement and development on the firmware after we succeed in the first implementation.

4.2.1 The development environment

To build OpenWrt, we use OpenWrt Buildroot. OpenWrt Buildroot is a set of Makefiles and patches that allows us to easily generate both a cross-compilation toolchain and a root filesystem for OpenWrt systems to be installed in our router. Here complilation toolchain is the set of tools for compiling code for our system. With OpenWrt written in C language, in our case the toolchain consists of a compiler(gcc [12]), binary utilities(binutils [14]) and a C standard library (GNU Libc [13]). The compilation toolchain comes with our system runs and generates code for the processor of our host system. In our case the processor type is x86. However, the OpenWrt system to be built for WRT54GL use brcm(broadcom) processor. That's why we need OpenWrt Buildroot with cross-compilation toolchain that can run on x86 system but generates code for brcm chips.

As we mentioned in section 3.2 , we plan to use Gargoyle interface [11] for OpenWrt development. As gargoyle is developed based on OpenWrt, it have makefiles associated with OpenWrt Buildroot tools that automatically build target system image from source code for us. Firstly, we need a Linux/GNU distribution, either a standalone installation or a separate system running in a virtual environment. In our case, we use Ubuntu 11.04 [28]. The we need to install all packets and libraries necessary for compiling and building work. This could be done by running below simple command for a 64 bit linux system:

sudo apt-get install build-essential asciidoc binutils bzip2 gawk gettext git libncurses5-dev libz-dev patch unzip zlib1g-dev subversion flex uglifyjs

4.2.2 Check out and build from code

After we set up the environment, it's time to check out the source and simply build one image. First we use svn [2] to check out the gargoyle source code:

svn checkout http://svn.github.com/ericpaulbishop/gargoyle.git

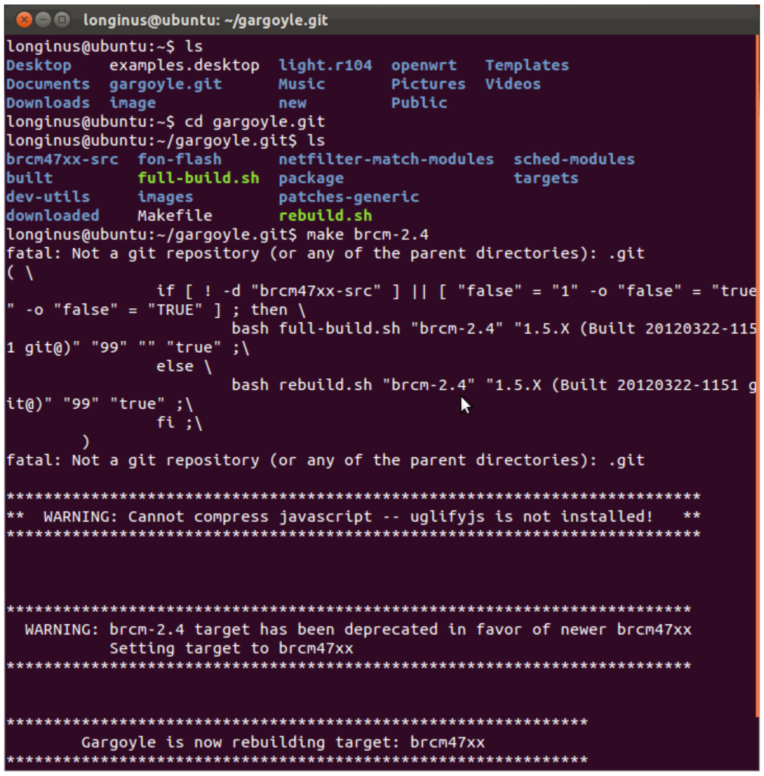
Under the gargoyle.git packet we have a makefile with two related shell files as full-build.sh and rebuild.sh. The first time we run command 'make' will take full-build actions that include checkout the stable version of OpenWrt source code and build the firmware based on OpenWrt with gargoyle web interfaces and functions. However, note that for different Linux kernels for different processor, each platform require more than 1.6GB of disk space to put the source code when compiling and building. The basic command 'make' will lead to a full build of all kinds of platform and require a really large disk space. Actually, our first building attempt failed with 'not enough disk space' error. So we should specify the target platform to save the disk space and reduce building time. In our case, targer router is using broadcom chips so the platform should be brcm-2.4 or brcm47xx, which made the build command like this:

make brcm-2.4

Then we can see lots of information flooding cross the screen to state the current progress shown in Figure 4.2 and after 3,4 hours (that's because we need to check out several GB of OpenWrt source code and save it to local memory at the first time) we can get the newly built images.

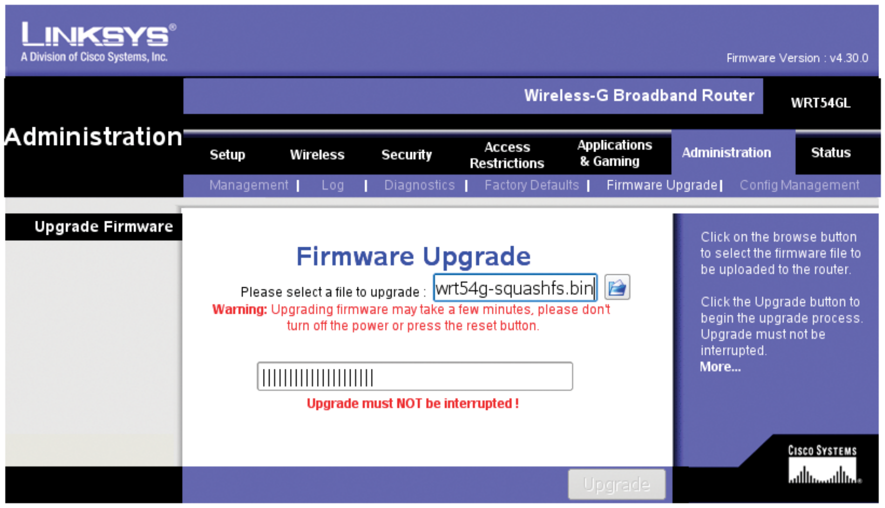
4.2.3 Firmware upgrade and router recovery

We can find all built images in Gargoyle.git/image. Here we pick out the image fit for different routers, in our case we pick 'wrt54g-squashfs.bin' for WRT54GL v1.1 router. If upgrade from OpenWrt, we can use another image named 'brcm47xx-squashfs.trx'. Most router firmware have the interface that allow user to upgrade the built firmware, for WRT54GL case, it is shown in Figure 4.3.



```
longinus@ubuntu:~$ ls
Desktop     examples.desktop  light.r104  openwrt   Templates
Documents   gargoyle.git      Music       Pictures  Videos
Downloads   image             new         Public
longinus@ubuntu:~$ cd gargoyle.git
longinus@ubuntu:~/gargoyle.git$ ls
brcm47xx-src  fon-flash     netfilter-match-modules  sched-modules
built         full-build.sh package                  targets
dev-utils     images        patches-generic
downloaded    Makefile      rebuild.sh
longinus@ubuntu:~/gargoyle.git$ make brcm-2.4
fatal: Not a git repository (or any of the parent directories): .git
( \
                if [ ! -d "brcm47xx-src" ] || [ "false" = "1" -o "false" = "true
" -o "false" = "TRUE" ] ; then \
                        bash full-build.sh "brcm-2.4" "1.5.X (Built 20120322-115
1 git@)" "99" "" "true" ;\
                else \
                        bash rebuild.sh "brcm-2.4" "1.5.X (Built 20120322-1151 g
it@)" "99" "true" ;\
                fi ;\
        )
fatal: Not a git repository (or any of the parent directories): .git

*************************************************************************
**  WARNING: Cannot compress javascript -- uglifyjs is not installed!   **
*************************************************************************



************************************************************************
  WARNING: brcm-2.4 target has been deprecated in favor of newer brcm47xx
           Setting target to brcm47xx
************************************************************************


*************************************************************
        Gargoyle is now rebuilding target: brcm47xx
*************************************************************
```

Figure 4.2: Building process in Ubuntu


LINKSYS
A Division of Cisco Systems, Inc.
Firmware Version : v4.30.0
Wireless-G Broadband Router
WRT54GL
Administration
Setup
Wireless
Security
Access Restrictions
Applications & Gaming
Administration
Status
Management | Log | Diagnostics | Factory Defaults | Firmware Upgrade | Config Management
Upgrade Firmware
Upgrade
Cisco Systems

Firmware Upgrade

Please select a file to upgrade : wrt54g-squashfs.bin

Warning: Upgrading firmware may take a few minutes, please don't turn off the power or press the reset button.

Upgrade must NOT be interrupted !

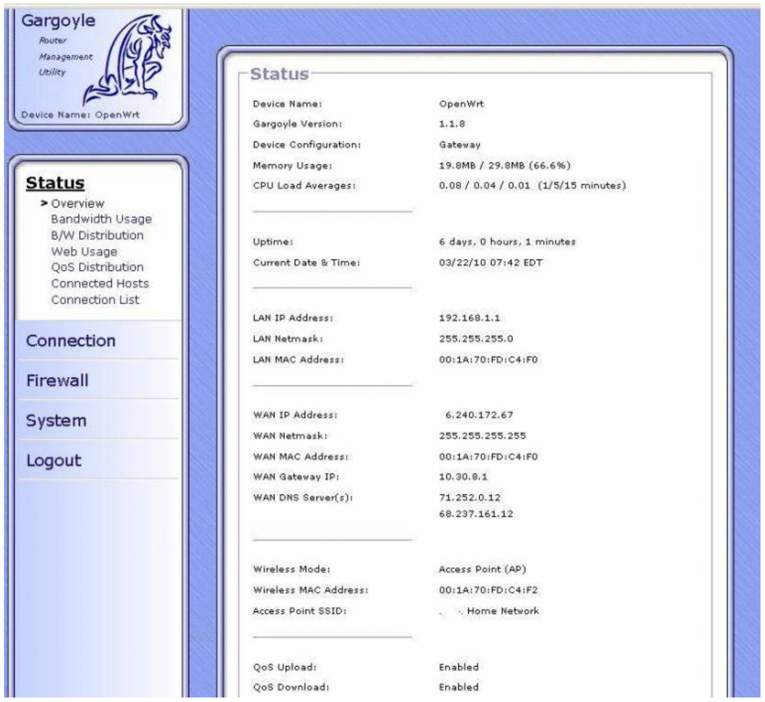
Click on the browse button to select the firmware file to be uploaded to the router.

Click the Upgrade button to begin the upgrade process. Upgrade must not be interrupted.
More...

Figure 4.3: Firmware upgrade

It is very important NOT to cut off the power of the router or disconnect between router and our client devices during the upgrade. After the router rebooted, we can see the gargoyle web interface as shown in Figure 4.4.

Sometimes there will be problems caused by configuration error that make it impossible to ac-



Status

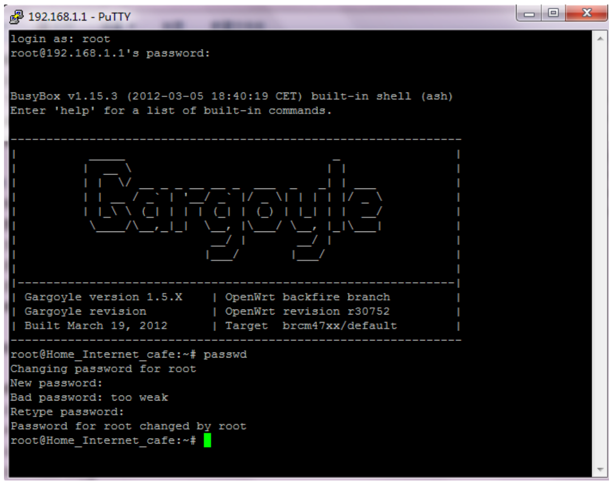
Device Name:	OpenWrt
Gargoyle Version:	1.1.8
Device Configuration:	Gateway
Memory Usage:	19.8MB / 29.8MB (66.6%)
CPU Load Averages:	0.08 / 0.04 / 0.01 (1/5/15 minutes)
Uptime:	6 days, 0 hours, 1 minutes
Current Date & Time:	03/22/10 07:42 EDT
LAN IP Address:	192.168.1.1
LAN Netmask:	255.255.255.0
LAN MAC Address:	00:1A:70:FD:C4:F0
WAN IP Address:	6.240.172.67
WAN Netmask:	255.255.255.255
WAN MAC Address:	00:1A:70:FD:C4:F0
WAN Gateway IP:	10.30.8.1
WAN DNS Server(s):	71.252.0.12 68.237.161.12
Wireless Mode:	Access Point (AP)
Wireless MAC Address:	00:1A:70:FD:C4:F2
Access Point SSID:	Home Network
QoS Upload:	Enabled
QoS Download:	Enabled

Figure 4.4: Upgrade OpenWrt firmware with Gargoyle interface

cess the router via the web interface. In our case when upgrade the firmware from normal OpenWrt to Gargoyle, I keep the default settings of OpenWrt, which leave the default password as empty. But Gargoyle don't accept an empty password, it keeps warning with "invalid password" and keeps me out from the web interface. In the other hand the SSH (Secure Shell) for router is also not accessable. To recover this or similar problems, we can us OpenWrt's failsafe recover mode:

1. Install Wireshark on computer
2. Configure our computer with static IP 192.168.1.2
3. Connect computer to a LAN port on the router
4. Start Wireshark and monitor the LAN connection
5. Powerup the router, when wireshark shows the first packet from 192.168.1.1 immediately press and hold the reset button on the back of the router for three seconds.
6. Router power LED should be flashing quickly. (Failsafe mode)
7. Use a command prompt to telnet the router via "telnet 192.168.1.1"
8. Use the command prompt of router to type command like "firstboot" or reset password via "passwd"
9. Reboot the router to apply the change

Figure 4.5 shows how the telnet for Openwrt works.



```
login as: root
root@192.168.1.1's password:


BusyBox v1.15.3 (2012-03-05 18:40:19 CET) built-in shell (ash)
Enter 'help' for a list of built-in commands.

---------------------------------------------------------------
|                                                             |
|          _____                             _                |
|         |  __ \                           | |               |
|         | |  \/ __ _ _ __ __ _  ___  _   _| | ___           |
|         | | __ / _` | '__/ _` |/ _ \| | | | |/ _ \          |
|         | |_\ \ (_| | | | (_| | (_) | |_| | |  __/          |
|          \____/\__,_|_|  \__, |\___/ \__, |_|\___|          |
|                          __/ |       __/ |                  |
|                         |___/       |___/                   |
|                                                             |
|-------------------------------------------------------------|
| Gargoyle version 1.5.X     | OpenWrt backfire branch        |
| Gargoyle revision          | OpenWrt revision r30752        |
| Built March 19, 2012       | Target  brcm47xx/default       |
---------------------------------------------------------------
root@Home_Internet_cafe:~# passwd
Changing password for root
New password:
Bad password: too weak
Retype password:
Password for root changed by root
root@Home_Internet_cafe:~#
```

Figure 4.5: Router recovering using telnet

TFTP method A more stable advanced way of firmware updating is the TFTP method. It is base on the bootloader functionality of the router. When router boots it runs a bootloader, which is used to perform basic system initialization along with validating and loading the firmware. It is like the BIOS of the device handling hardware before control is passed over to the operating system loaded from the firmware. When the firmware fail to pass a cyclic redundancy check (CRC) done by the bootloader, the bootloader will presume the firmware is corrupt and wait for a new firmware to be uploaded over the network. And we can enable the task of waiting for a new firmware to be uploaded every time when router boot by typing command lines in router's ssh or telnet interface:

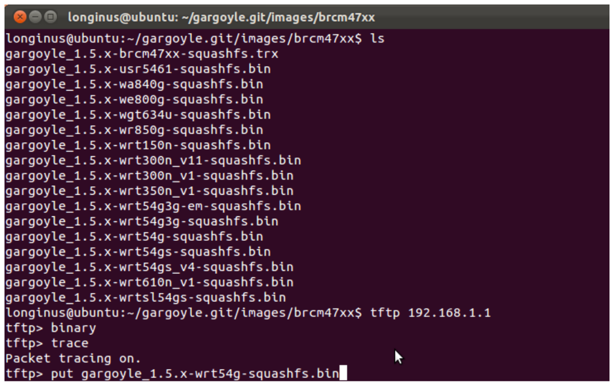
nvram set boot_wait=on
nvram set boot_time=10
nvram commit && reboot

The router will reboot automatically after execute the command. Then every time we boot the router, it will wait for 10 seconds to allow a new firmware to be uploaded into the memory and run operating system based on the new firmware. We use TFTP method to upload the new firmware. TFTP stands for Trivial File Transfer Protocol as a file transfer protocol generally used for automated transfer of configuration files between machines in a local environment. We set our computer as a tftp client to upload new firmware to the router. In this way we can avoid lots of problems caused by firmware errors that may even not provide interface for us to update firmware normally. The procedure of upload firmware after bootwait time is set is shown as following steps:

1. unplug the power to the router

2. Configure our computer with static IP 192.168.1.2
3. Connect computer to a LAN port on the router
4. start a tftp client
5. set send mode and file to be sent (in our case the new firmware)
6. Powerup the router, while having the tftp client running and constantly probing for a connection
7. the tftp client will receive an ack from the bootloader and starts sending the firmware
8. wait for the router booting with newly updated firmware

In step 5 we connect to router via the command 'tftp 192.168.1.1' and set the send model as 'binary' and 'trace' to divide target file into data blocks and trace each packet sent with ack to ensure the sending task succeed. Figure 4.6 shows the tftp method in ubuntu to update target firmware.



```
longinus@ubuntu:~/gargoyle.git/images/brcm47xx$ ls
gargoyle_1.5.x-brcm47xx-squashfs.trx
gargoyle_1.5.x-usr5461-squashfs.bin
gargoyle_1.5.x-wa840g-squashfs.bin
gargoyle_1.5.x-we800g-squashfs.bin
gargoyle_1.5.x-wgt634u-squashfs.bin
gargoyle_1.5.x-wr850g-squashfs.bin
gargoyle_1.5.x-wrt150n-squashfs.bin
gargoyle_1.5.x-wrt300n_v11-squashfs.bin
gargoyle_1.5.x-wrt300n_v1-squashfs.bin
gargoyle_1.5.x-wrt350n_v1-squashfs.bin
gargoyle_1.5.x-wrt54g3g-em-squashfs.bin
gargoyle_1.5.x-wrt54g3g-squashfs.bin
gargoyle_1.5.x-wrt54g-squashfs.bin
gargoyle_1.5.x-wrt54gs-squashfs.bin
gargoyle_1.5.x-wrt54gs_v4-squashfs.bin
gargoyle_1.5.x-wrt610n_v1-squashfs.bin
gargoyle_1.5.x-wrtsl54gs-squashfs.bin
longinus@ubuntu:~/gargoyle.git/images/brcm47xx$ tftp 192.168.1.1
tftp> binary
tftp> trace
Packet tracing on.
tftp> put gargoyle_1.5.x-wrt54g-squashfs.bin
```

Figure 4.6: Firmware update using tftp method

4.2.4 Key files related to user interface

We now succeed in setting up develop environment, compiling code and building the image. The next step is to add our simple own page into the system.

The idea of gargoyle interface for OpenWrt is to do computation on the client side to make operations faster and reduce delay. Majority of gargoyle is implemented as Javascript. Source files for the javascript are saved at /www/js directory to provide functions. The server-side, which is the router in our case, will do necessary scripting using haserl [18] , a very lightweight utility that enables the embedding of linux shell script into html files. Those html/haserl scripts are saved as *.sh files in the /www directory which manage the display of the interface with actions done by functions described in related javascript files as *.js. Several large pages that have complex display structure have template files written in html and saved as *.css, which can be directly used with the features defined in it. For the web page structure, as header/footer/menu code is generated from the config file in directory /etc/config/gargoyle. This file is written in C because shell scripts execute really slow for page loading. These four kinds of files contain the key code of the gargoyle web user interface. The relationship of the different files are describe in Figure 4.7.

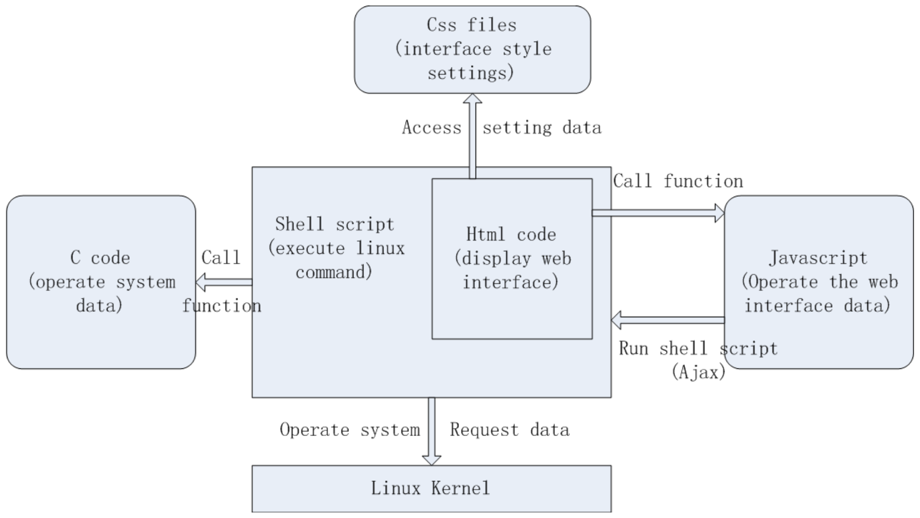


Figure 4.7: Relationship of different programming code

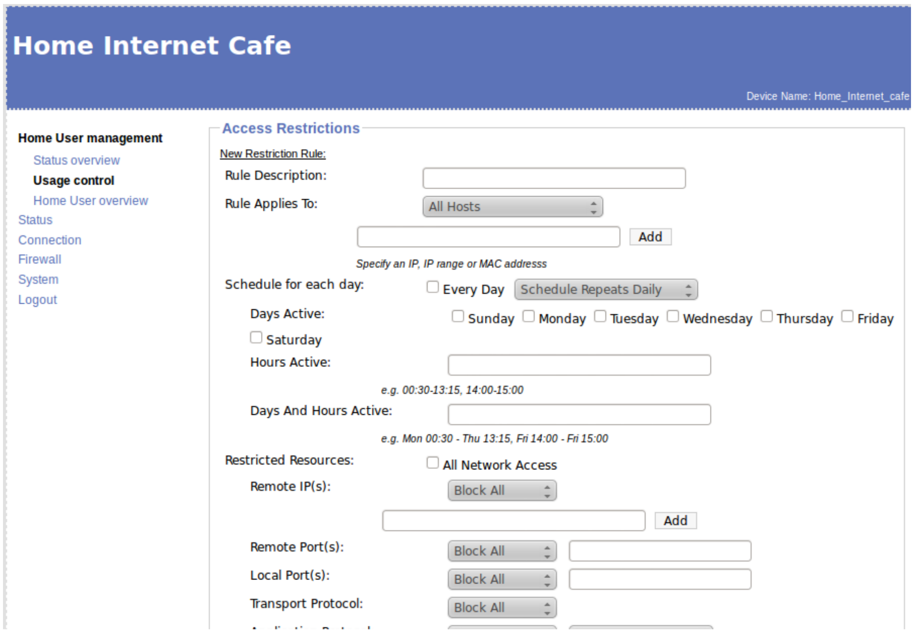
As shown in the figure. Shell script is the key files in the system since all the linux commands operating our system are executed here. Usually basic commands like requesting system data are written in it. And for large amount of system data operating, we build functions written in C languages and use them in the shell script then. The user interface run shell script from a web page via Apache CGI [3]. Apache CGI allows our script in cgi-bin directory treated as application and run by the router server when requested by our user. During the build process shell script is put in cgi-bin directory. Then user is able to execute them from a web page. When user access the shell files, to display the user interface, we type html codes in the shell script using the following code as the header in shell script.

```
 echo "Content-type: text/html"
echo ""
```

CSS known as Cascading Style Sheets is the widely used style sheet language used by us to describing our web pages' display formatting. Also Javascript is a widely used scripting language. We use it to handle the functions and operations inside the interface web pages. In some cases like user logging in, the data handling by javascript from the web interface input need to send to linux system and get request, we use Ajax method to send the commands with data and run them in shell script. Ajax stands for Asynchronous JavaScript and XML which allow us to send data to and retrieve data from our router asynchronously without interfering with current web page's display. With all these different kind of files written in different programming languages we are able to provide the whole system's features and interface.

4.2.5 Implementation result

We make small changes in config file to build our new menu with home Internet cafe related fucntions and change the page theme via the *.css template files to give a different display using pure color without pictures to make the page load faster and reduce the image size. The result of our implementation is shown in Figure 4.8.


Home Internet Cafe

Device Name: Home_Internet_cafe

Home User management
- Status overview
- Usage control
- Home User overview
- Status
- Connection
- Firewall
- System
- Logout

Access Restrictions

New Restriction Rule:

Rule Description:	
Rule Applies To:	All Hosts
	Add
	Specify an IP, IP range or MAC addresss
Schedule for each day:	☐ Every Day Schedule Repeats Daily
Days Active:	☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday
Hours Active:	
	e.g. 00:30-13:15, 14:00-15:00
Days And Hours Active:	
	e.g. Mon 00:30 - Thu 13:15, Fri 14:00 - Fri 15:00
Restricted Resources:	☐ All Network Access
Remote IP(s):	Block All
	Add
Remote Port(s):	Block All
Local Port(s):	Block All
Transport Protocol:	Block All

Figure 4.8: Web user interface after simple test implementing

4.3 Key functionality design

4.3.1 UCI system and UCI related functions

The UCI stands for Unified Configuration Interface and is set up to centralize all the configuration related to our system based on OpenWrt firmware [20] . With UCI system we do not need to bother different kinds of config-file lying in different folders. Instead we only change the UCI configuration files and the system can get the stored settings and work based on them.

Our OpenWrt configuration settings are split into several uci files located in the route /etc/-config/directory. We can edit the configuration uci files with a text editor or with the command line utility program 'uci' or through various programming APIs like shell script and C language. Among many different uci files we focus on the two uci configure files related to firewall and system.

To show how uci system works, we give out a small part of uci system file below as an example:

```
 config display display
option management 'Home User management'
option management_hosts1 'Home User overview'
option management_homeuser 'Home User control'
config scripts scripts
option management_hosts1 'hosts.sh'
option management_homeuser 'Homeusr_control.sh'
```

In our example we pick out a small part of the menu display configuration of our interface. The uci configuration files usually consist of different config statements named sections with one or more statements called option defining useful values. Here we create two sections via 'config': 'display' and 'scripts', both with their section names and types the same. The type of the section can be used by us to decide how to treat the options it contained in the further implementation. Then we add options of each section via 'option' to store different values with a label name. Later in the menu display part, we write functions that get the target label's value in display section to get the section name to display in the menu and set related interface scripts via get the target label's value in the scripts section to run when target label is chosen by users.

Common way to configure uci files is command line utility works in the ssh interface shown in our router recovery subsection 4.2.3. In ssh interface we type command lines like 'uci commit packagename' to save target package settings, 'uci add' to create sections, 'uci set' to manage options' value and 'uci get' to get the value we need. However, when we provide web interface to our users, it's not able to ask them typing these commands to operate the system works for user requests. So we develop a set of uci related functions in our C code and Javascript code to realize different actions and store user settings.

As we want to provide user identify functions and time limitation functions, different lists of values need to be set by our user and stored for further use. UCI system is very convenient but setting up sections and options via command lines will be complex and not so user friendly: it's not possible to ask your user to learn a new command syntax and act like a programmer typing code to manage the system. Then to provide graphical interface for users to add their settings in the system and store the settings as uci files to be used by the system, we combined uci command lines with C and Javascript functions. In the C code part we have the uci.h library which contains functions directly reading the values in uci files. And our user interface functions are written in javascript. So we call C functions in the shell script to get the data in uci files, then use javascript functions to display and operate uci data based on users' operations with the graphical interface and change the settings.

The values in uci files read by c functions are stored as a set of array objects with section and option names as label and type and option values as data stored. Then these array objects can be used by the javascript functions to display the settings to our users and allow them to change and add settings. Below are the functions written by us to manage the uci setting data.

UCI related functions: In default the values in the uci files read by c functions are all stored in an object named uciOriginal. Values are stored in uciOriginal based on the files, sections and options they belong to. For example, we operate the activehours setting of target users in a rule via access uciOrignal.firewall.rules1.actvehours. The uci related functions use the 'this' method of javascript which allow us to write functions as this.function and apply them to target object via replace the object name with 'this'. So the below functions we can replace 'uci' with different uci object names like 'uciOriginal' to operate target objects.

uci.clone This function clone all the data in target uci object and return the stored values as a new uci object. We always use this functions inside the shell scripts directly after we get the default uciOriginal object with the code

var uci = uciOriginal.clone

to get a new uci object and let users to operate the uci object instead of uciOriginal object. We will introduce the reason in the uci.getScriptCommands function.

uci.get This function access target sections/options in uci file and return its value for further use. To make it easier to use this function, we make the variables operated by this function similar to the route of uci values stored. The functions works like this:

variable = uci.get(target file name, target section name, target option name)

uci.set This is a basic function that stores the values changed or added by users in the graphical interface. It works the same to uci.get as

uci.set(target file name, target section name, target option name, changed/added value)

to store an option value, and

uci.set(target file name, target section name, changed/added type)

to create a new section with the specific type.

uci.remove This function is used for removing target sections or options in uci settings. It also works with target values' access route:

uci.remove(target file name, target section name, target option name)

uci.getAllSectionsOfType We mentioned before that each section in the uci setting files have a 'type' value. The type of the section can be used by us to decide how to treat the options it contained. When we set up a section using uci.set we give out type value. In our system the type usually is 'homeuser' or 'restriction_rules'. We can take the 'homeuser' type as an example. In our code indentify system when we get the homeuser name and password typed by our user, we need to access all the homeuser sections of the uci system file and verify the homeuser name and password options stored in them to check whether our user typed the right name and password to login and access the Internet. We use uci.getAllSectionsOfType function as

uci.getAllSectionsOfType(target file name, target section type)

and in this case the code should be 'homeuser = uci.getAllSectionsOfType(system, homeuser)'. Note that this function returns a set of sections as array object, so the 'homeuser' should be a predefined array object via 'var homeuser = []'. Then we can easily verify the user input with all homeuser sections with the 'for' loop.

uci.getScriptCommands This is the key function of our uci related functions. As we introduced, we get a new array object via uciOriginal.clone named 'uci' as a copy the array object 'uciOriginal' read by c code from uci files. Then we use different uci related functions to allow user change, delete or add values in the system setting interface and store the changes in 'uci' object. All the changes are only stored in the 'uci' object belong to javascript functions in this case. User settings are not even get stored in the uci files, and not applied in the system either. Here we use getScriptCommands to generate the scripts of uci command lines to be executed in shell script via the comparison of user changed 'uci' object and the original 'uciOriginal' object:

command = uci.getScriptCommands(uciOriginal)

We mentioned that the 'uci' object has been changed by user via the graphical interface. The function getScriptCommands generate the scripts based on the differences of changed 'uci' object and the 'uciOriginal'. The command 'uci add sectionname' is generated when new section exists in 'uci' compare to 'uciOrignial'. The command 'uci set options value' is generated for new options added in different sections. The command 'uci delete sectionname' or 'uci delete sectionname options' is generated when sections or options in 'uciOriginal' are deleted in 'uci'. All these generated functions are returned by the function as strings stored in the command variable. And to save the uci command lines operation result, we always add 'uci commit' at the end of the command via command.push() function. Till now the uci command lines generated are stored in a variable belong to JavaScript functions and still not applied by the system. To run these command lines in the system, we export them to shell script and execute them there via the Ajax method.

Ajax for uci set We introduced Ajax as Asynchronous JavaScript which is used by javascript to send request and get data associated with shell scripts in the code relationship Figure 4.7. To apply user settings, we use Ajax for sending to be executed uci command lines and get respond as below:

```
var commands = uci.getScriptCommands(uciOriginal);
var param = getParameterDefinition("commands", commands);
var stateChangeFunction = function(req)
{
if(req.readyState == 4)
{
uciOriginal = uci.clone();
Getrulesfromuci(onedituser);
alert("Changes applied. The rule will be active when next time target user log in.");
}
}
runAjax("POST", "utility/run_home_commands.sh", param, stateChangeFunction);
```

We use a fuction to compile the string variable commands into a set of strings that can be directly used by shell scripts. Then we write the 'stateChangeFunction' which describe javascript actions after we get successful respond from the system. The actions include saving the applied uci configuration as the default 'uciOriginal' object to allow further user edit after settings applied, displaying the changed configuration and informing user that settings are applied. Then we use the runAjax function to call the shell script run_home_commands.sh and send commands. When commands successfully executed, we get a respond that change the variable 'req.readySate' into value 4. Then we provide user interface changes based on the functions we wrote in 'stateChangeFunction' to finish applying the user settings.

After Ajax function successfully executed and our functions get the respond to inform users setting applied, we get the user settings and data stored in the system which can be used when target functions for user control are called. All the important data in our home Internet cafe system like home user data, time restriction settings, user login status is all stored and managed under this UCI system.

4.3.2 Time restriction system

Linux firewall and Iptables

The key feature of our home cafe system is to handle users' Internet usage. Most of the routers using linux kernel realize user restrictions by iptables [21] . Iptables is an application program for configuring the chains and rules user set for the linux kernel firewall to manage the data packets. In our scenario of home user system with all devices connected to router to get Internet access, all users' data packets go through router's linux kernel where we can use iptables to add different rules to accept or drop target packets.

Packets enter router firewall and hit the hardware first. Then before being sent to local applications or forward to target host, packets should go through a series of steps in the linux kernel. A set of steps are named chains with different kinds of rule tables like raw, mangle, nat and filter. Those tables are used for different kind of packets handling while in our system we focus on the filter table. As its name states, filter table do all the filtering for data packets. We can manage user's Internet use by adding rules in the filter table of iptables. Figure 4.9 shows how iptables handle the packets. Here we have three different chains shown as INPUT, OUTPUT and FORWARD. Packets with local host as destination go through INPUT chain. Packets with local host address as source will go through OUTPUT chain. And packets destined for another host on another network will go through the FORWARD chain. In our use case, handling FORWARD chain is enough to limit users' access to the Internet.

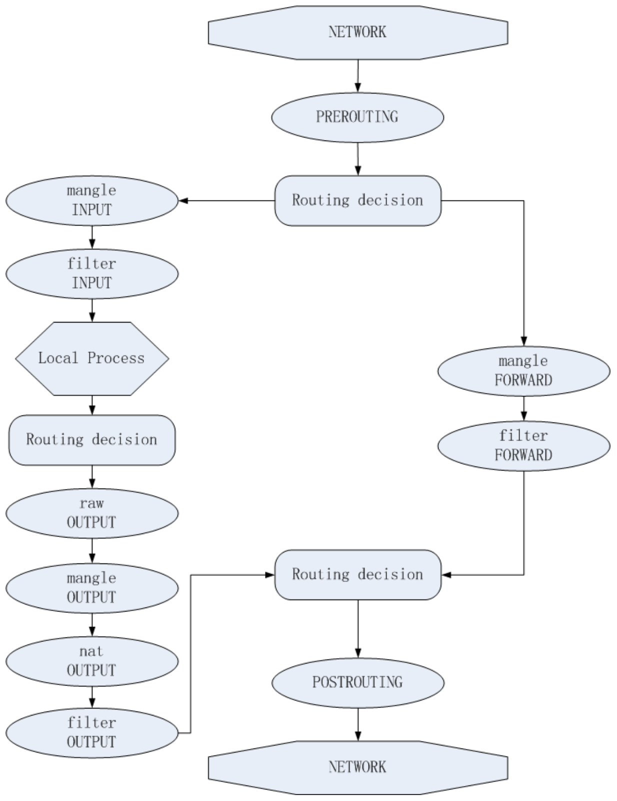


Figure 4.9: Packets go through chains of Iptables

As we mentioned, the chain as a set of steps with different tables handling our packets and we focus on the filter table. Rules are added in the filter table and stored in a fixed order. When a packet enters a chain such as the INPUT chain in the filter table, it will go down from rule to rule verifying the packet until the packet match the rule and is operated by the rule command or the main chain ends. If main chain ends, default policy of the built-in chain will be applied. Figure 4.10 shows how the rule chains in our system are built.

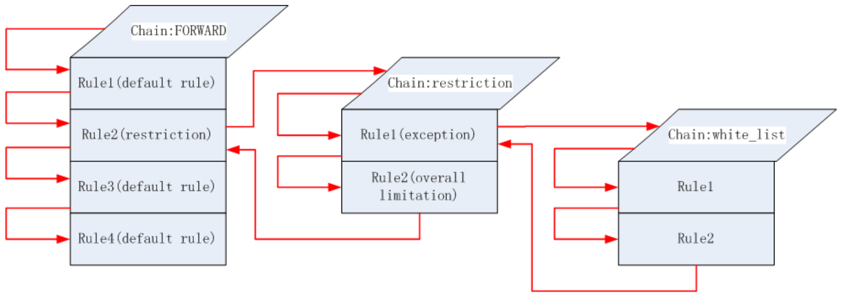


Figure 4.10: Time restriction chains and rules in FORWARD filter

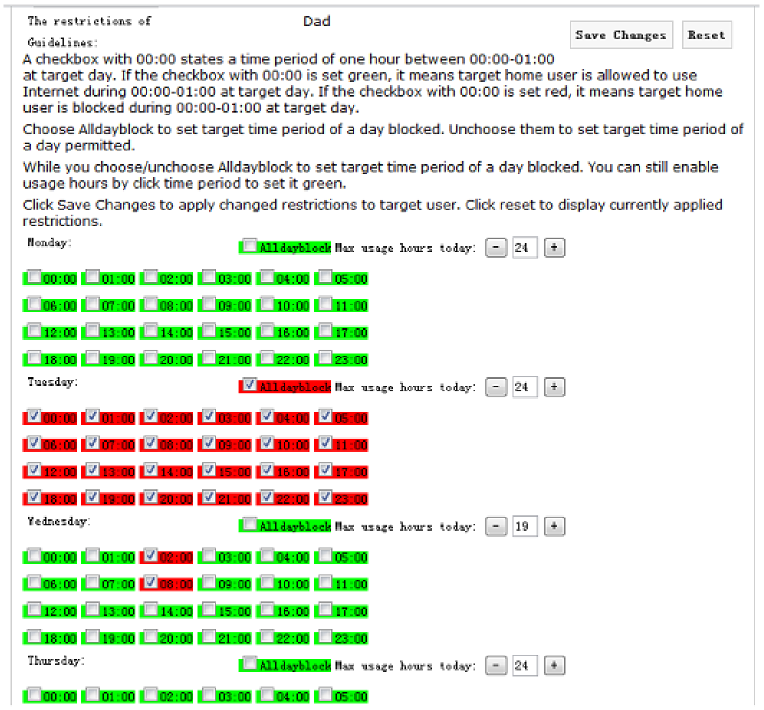
When user packets go through the FORWARD chains of Iptables in our system, it first go through some normal router setting rules and reach the restriction rule, which points to a new chain that contains different rules. As our figure shown, packets will then go inside the restriction chain. The first rule in restriction chain is the whitelist rule which also is a new chain containing other rules. Then packets will go inside the whitelist chain and be verified by different rules. We mention that packets will stop at the rule it matches and be operated by target rules. So if no rules matches, it will return to the the formal chain after current whitelist chain ends and go through the left rules in restriction chain. Then if it is still not operated it will return to the FORWARD chain and go through the left rules and then get managed by the system.

Home Internet cafe Iptables In our system, the rule in restriction chain is set as:

iptables -A FORWARD -p tcp ! -s 192.168.1.1 -j REJECT

This code describe a rule that all the tcp packets will be rejected if their destination addresses are not our router host address: 192.168.1.1. With this rule exists, we manage to allow every user in our system only to access the web interface when connected to the router. Then to allow home users to access the Internet. We add rules in the whitelist chain as exceptions to allow user packets forward by the system instead of get rejected. Those rules in whitelist are set based on home user management settings and the code is set up via functions combined in different program languages. Now we will introduce how those rules are made and how user restrictions with total usage time limitation are realized. Figure 4.11 shows the interface of home user restriction settings.

The restriction settings are given as time period blocks of every day with green as accepted and red as blocked to make user configure the restriction rules straight forward. We manage the background color of checkboxes via the mouse events functions of Javascript include onMouseOver,



The restrictions of Dad

Save Changes Reset

Guidelines:

A checkbox with 00:00 states a time period of one hour between 00:00-01:00 at target day. If the checkbox with 00:00 is set green, it means target home user is allowed to use Internet during 00:00-01:00 at target day. If the checkbox with 00:00 is set red, it means target home user is blocked during 00:00-01:00 at target day.

Choose Alldayblock to set target time period of a day blocked. Unchoose them to set target time period of a day permitted.

While you choose/unchoose Alldayblock to set target time period of a day blocked. You can still enable usage hours by click time period to set it green.

Click Save Changes to apply changed restrictions to target user. Click reset to display currently applied restrictions.

Monday: ☐ Alldayblock Max usage hours today: - 24 +

☐ 00:00 ☐ 01:00 ☐ 02:00 ☐ 03:00 ☐ 04:00 ☐ 05:00
☐ 06:00 ☐ 07:00 ☐ 08:00 ☐ 09:00 ☐ 10:00 ☐ 11:00
☐ 12:00 ☐ 13:00 ☐ 14:00 ☐ 15:00 ☐ 16:00 ☐ 17:00
☐ 18:00 ☐ 19:00 ☐ 20:00 ☐ 21:00 ☐ 22:00 ☐ 23:00

Tuesday: ✓ Alldayblock Max usage hours today: - 24 +

✓ 00:00 ✓ 01:00 ✓ 02:00 ✓ 03:00 ✓ 04:00 ✓ 05:00
✓ 06:00 ✓ 07:00 ✓ 08:00 ✓ 09:00 ✓ 10:00 ✓ 11:00
✓ 12:00 ✓ 13:00 ✓ 14:00 ✓ 15:00 ✓ 16:00 ✓ 17:00
✓ 18:00 ✓ 19:00 ✓ 20:00 ✓ 21:00 ✓ 22:00 ✓ 23:00

Wednesday: ☐ Alldayblock Max usage hours today: - 19 +

☐ 00:00 ☐ 01:00 ✓ 02:00 ☐ 03:00 ☐ 04:00 ☐ 05:00
☐ 06:00 ☐ 07:00 ✓ 08:00 ☐ 09:00 ☐ 10:00 ☐ 11:00
☐ 12:00 ☐ 13:00 ☐ 14:00 ☐ 15:00 ☐ 16:00 ☐ 17:00
☐ 18:00 ☐ 19:00 ☐ 20:00 ☐ 21:00 ☐ 22:00 ☐ 23:00

Thursday: ☐ Alldayblock Max usage hours today: - 24 +

☐ 00:00 ☐ 01:00 ☐ 02:00 ☐ 03:00 ☐ 04:00 ☐ 05:00

Figure 4.11: Time restriction setting interface

onMouseOut and onClick. These functions will create events in cases that mouse is passed over, taken off or clicked on an active checkbox. In our system these events are background color changes and checkbox checked or unchecked. We also have function that allow user set the total usage limitation per day counted by hours from 1 to 24. The interface of this function is shown as a input box with numbers that user can edit the number with the add and minus buttons or type a number by themselves. To avoid the wrong input like words instead of numbers in the settings, we add a small function within the input box using regular expression tester to verify and change the user input:

```
function testFieldnumber(field)
{
var regExpr = new RegExp("^\d*\.?\d*$");
if (!regExpr.test(field.value))
{
field.value = "24";
}
}
```

As we have mentioned, the UCI system in OpenWRT allows users to store their system settings and let functions to pick up those values and apply them into system settings. In our restriction set up page, when user clicks 'save changes' button, we will use the uci set method as we used in the code Identification system to store the user restriction rules and total limitation time settings for target home user. While all users in default settings are blocked from the Internet, we set up Iptable rules that allow user to access the Internet in specific time period based on the settings stored in the uci system. In the next subsection, we will introduce it in details how our time restriction system are applied during the home user login action in our code identify system.

4.3.3 Code Identify System

Another key feature of our home Internet cafe system is the code identify system. We would like to build a system that works like the Internet cafe where user type their user ID and password to get the access to the Internet. And the administrator, in our case it's parent account, can manage different users' Internet usage in the code identify system based on the logged in users' id instead of manage via ip or mac addresses. The overall code identify system authentication process is shown in Figure 4.12 and Figure 4.13.

As we mentioned, in default all users' data packets with a destination address different from the router ip 192.168.1.1 will be rejected. So user have to log in as home user to get the Internet access. The code identify home user authentication starts when users type their user ID and user password and click home user login button. The first task is verify user ID and password. Via the uci related functions, we get the home user ID options stored in the system uci file created by parent account. We first compare user ID. If a matched user ID is found in the uci file section options, we continue to compare target password with the password user typed. We will go to next step if both user ID and password match the same home user ID created and stored in our system. Else we will alert the user with error message and stop process.

After we pick out target home user ID that our user want to login as, we verify a set of status about target home user ID. The status is accessed via uci functions with home user ID as the section name to get options containing status value. First we verify the option with a boolean value states the user status as enabled or blocked. If target home user ID is blocked, we give out notification and stop the process. Else we will continue to verify other status. But before that we need current system time and date and the mac address of the user who is currently trying to login.

To get the current system time and date, we execute below commands in shell scripts:

```
current_day=$(date "+%a")
current_time=$(date "+%s")
echo "var currentDay = \"$current_day\";"
echo "var currentTime = \"$current_time\";"
```

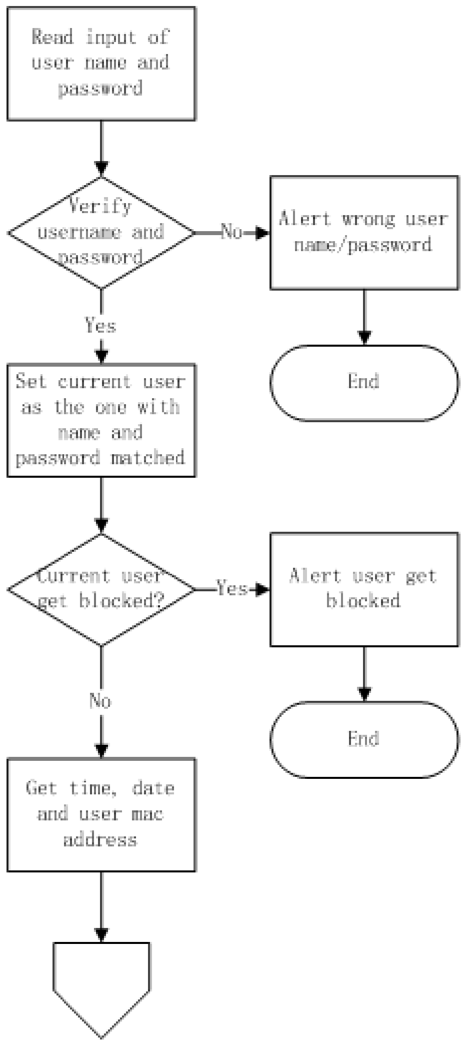

Read input of user name and password
Verify username and password
No
Alert wrong user name/password
End
Yes
Set current user as the one with name and password matched
Current user get blocked?
Yes
Alert user get blocked
End
No
Get time, date and user mac address

Figure 4.12: Code identify system authentication process part1

Here we use two different linux date command with different suffix. The '%a' suffix ask the system to return locales abbreviated weekday name like 'Sun'. And the '%s' suffix ask the system to return the total number of seconds since 1970-01-01 00:00:00 UTC, known as the unix epoch time. In either C or javascript we have specific functions that can transfer the epoch time into a human readable time form.

User's mac address is generated in javascript functions from the Address Resolution Protocol(ARP) data with the help of target user's ip address. First, we execute below commands in shell scripts:

echo "var connectedIp = \"$REMOTE_ADDR\";"
echo "arpLines = new Array();"
cat /proc/net/arp — awk '{print "arpLines.push(\""$0"\");"}'

After the command is executed we get the array object 'arpLines' that stores all connected users' data include ip address, mac address, host name, etc. At the same we get current user's LAN ip address via linux command and stored it in the variable 'connectedIP'. Then we write a function to pick out current user's mac address in the 'arpLines' array using current user's ip address, that is to find out the only list of user data containing target ip address and export the mac address in the list. The exported address is current user's mac address we need for further use.

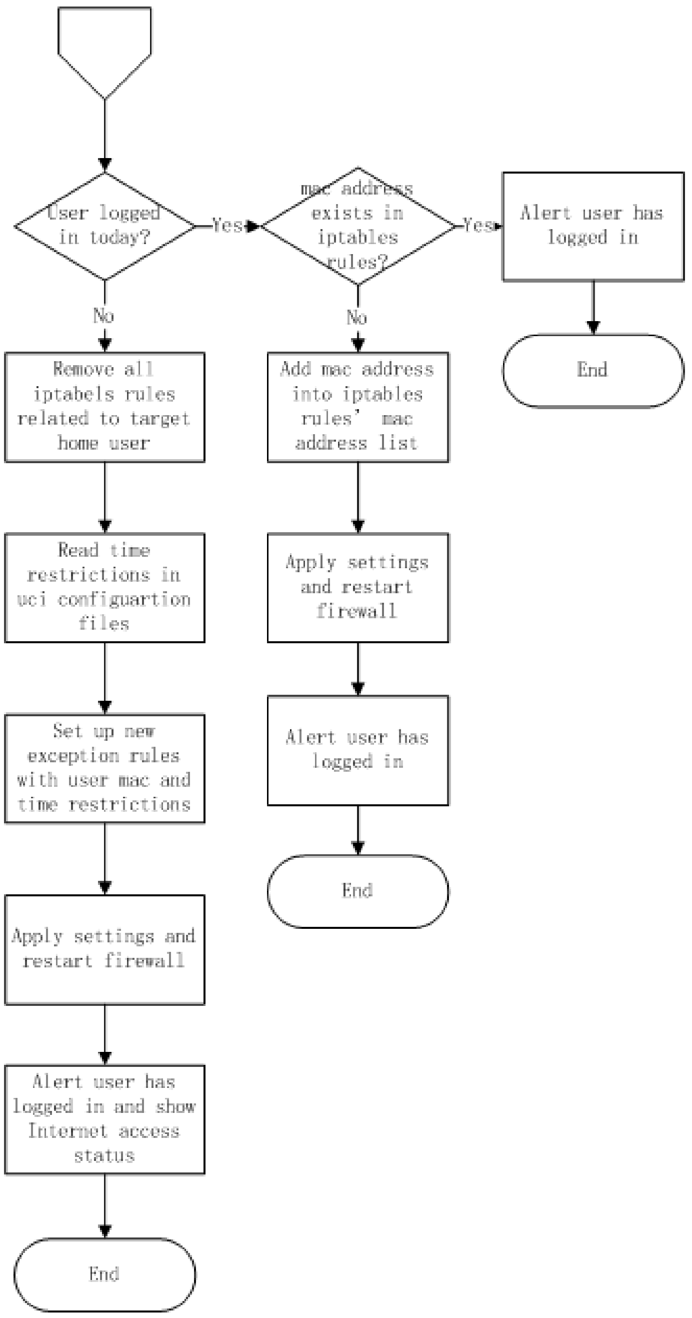

User logged in today?
Yes
mac address exists in iptables rules?
Yes
Alert user has logged in
End
No
Remove all iptabels rules related to target home user
Read time restrictions in uci configuartion files
Set up new exception rules with user mac and time restrictions
Apply settings and restart firewall
Alert user has logged in and show Internet access status
End
No
Add mac address into iptables rules' mac address list
Apply settings and restart firewall
Alert user has logged in
End

Figure 4.13: Code identify system authentication process part2

Then we verify another status of target home user ID, which describes whether the home user

ID has logged in the system today or not. The result of the verification will lead to two different main tasks in the further process steps based on the value. Every day a home user ID first login, it will store current day as the value of the uci option 'lastlogin'. Then we can distinguish target home user ID has logged in or not by comparing 'currentDay' variable and the value of uci option 'lastlogin' in target home user section. If they are the same, the process will go through the logged in steps, else we do first login steps of a day.

First let's go through the first login process in case that it's the first target home user ID login today. Since in default all users' tcp packets sent outside are blocked by the iptables rule, we provide user Internet access via generate exception rules based on the settings of time restriction system operated by parent account and apply them in the whitelist chain in iptables before the restriction chain to forward target users' packets before they get rejected. As shown in Figure 4.13, at first we will remove all the formal rules related to target home user ID regardless of whether there is related rule or not to avoid conflict and redundancy. Then we start to set up new exception rules based on settings from time restriction system via reading the uci configuration file.

In time restriction system subsection we introduced that in our interface user's access time is divided into time period blocks. Each block contain 1 hour. When storing the restriction time period of a weekday, we pick out the time period when user is allowed to access Internet and push the number into the active time array of target weekday. For example, when a user are allowed to access Internet during 18:00-19:00 at Monday, we add number '18' into Monday's active time array. While in the rules generation step, we first generate the hour of current time by transfering the unix epoch time with javascript functions. Then we compare current hour with each hour number in current day's active time array from uci files and start to generate exception rules when we find a number in the array equal to or large than current hour. Another important value is the total limitation time of current day. If the value is 0, no rules will be generated and user is totally blocked. When the total limitation time is not 0, we keep generating iptables rules in below form:

iptables -A FORWARD -p tcp –mac-source USER MAC ADDRESS –timestart xx:00:00 –timestop (xx+1):00:00 -j ACCEPT

The timestart and timestop value are the time period of 1 hour between active hour xx o'clock and xx+1 o'clock. This generating work stops either all active time applied in the rules or the number of rules is equal to target day's total limitation number. Note that if rule generations stop because of the total limitation and the timestop value of the last rule hour isn't 24, we need to add current minutes into the timestop value of the last rule as xx:current minutes:00 instead of (xx+1):00:00 to ensure user only get limited Internet surfing hours without extra minutes.

If uci option 'lastlogin' in target home user section shows it is not the first time user login, we take different actions. We have an option in home user section that stores related mac addresses with iptables exception rules applied at current day. So we verify whether the mac address of target user exists in the mac addresses list. If it exists, it means user has already logged in today. We will then inform the user that home user ID he have logged in and the time left for user to surf the Internet. If the mac addresses list doesn't contain target mac addresss, we re generating iptables rules based on existing rules via adding the new mac address in the '–mac-source' option to allow the new user device access the Internet with the formal time restriction settings.

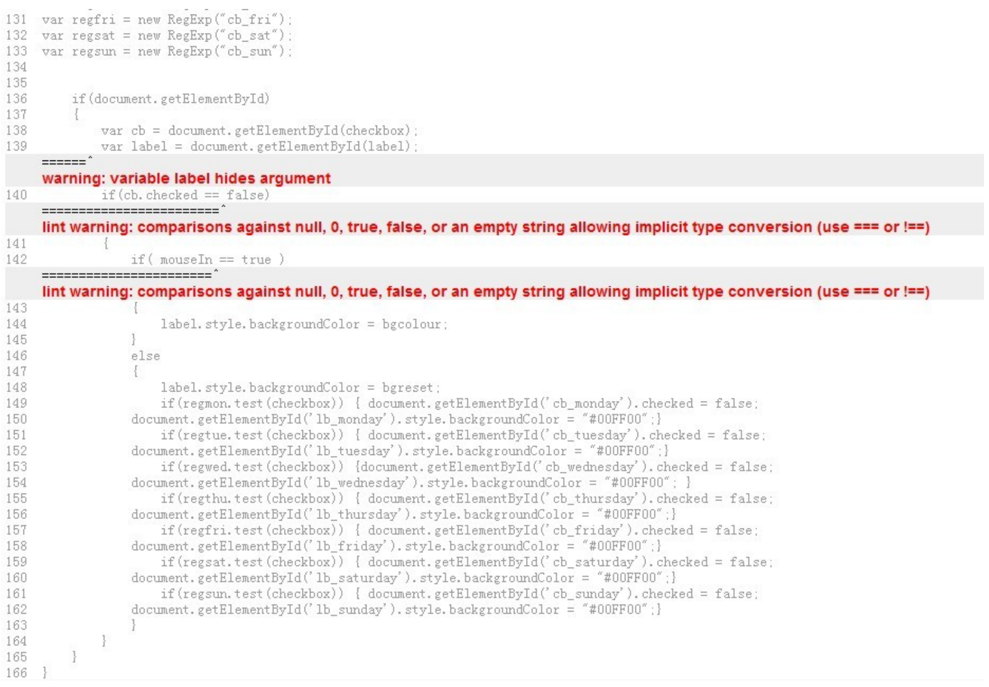
As the login process shows, our code identify system handles user mac/ip automatically which only ask users to type user ID and password. Also the parent account as rule maker easily manage Internet usage via set up restrictions based on home user ID. There is no need for them to type mac address or ip address by themselves which can be very confusing work for parents with less IT knowledge.

4.4 Debug tools

During our implement, debug tools are necessary in each small step to verify the code syntax and functions. Shell script and html code errors will lead to web page display errors which can be easy to notice and fixed. But bugs of javascript functions are not so easy to find as web page display normally but values may not be calculated, which lead to the result that user operations can't get feedback. But the javascript syntax error will lead to no feedback error also. Then we use special debug tools to fix these bugs.

4.4.1 Javascript lint

Many JavaScript implementations do not warn against questionable coding practices. The way to verify the code is to run it as part of a page in web browser. But in our case things are different. Our javascript code is related to shell script which need to be run in the router linux system. However, to get the system shown, we must build target firmware via compiling all our code and update the built firmware to our router. After that we have to wait for a period of time when router is rebooting. All the whole complex work will be useless if a small syntax error exists in the javascript code. So we need a debug tool that can allow us to check all our JavaScript source code for common mistakes without actually running the script or opening the web page. The Javascript syntax check tool 'Javascript lint' fits our requirement [25]. And it holds an advantage as it is based on the JavaScript engine for the Firefox browser which can provide a robust framework that can not only check JavaScript syntax but also examine the coding techniques used in the script and warn against questionable practices. Figure 4.14 shows how Javascript lint works with error alert.



```
var regfri = new RegExp("cb_fri");
var regsat = new RegExp("cb_sat");
var regsun = new RegExp("cb_sun");


    if(document.getElementById)
    {
        var cb = document.getElementById(checkbox);
        var label = document.getElementById(label);
======^
warning: variable label hides argument
        if(cb.checked == false)
========================^
lint warning: comparisons against null, 0, true, false, or an empty string allowing implicit type conversion (use === or !==)
        {
            if( mouseIn == true )
=======================^
lint warning: comparisons against null, 0, true, false, or an empty string allowing implicit type conversion (use === or !==)
            {
                label.style.backgroundColor = bgcolour;
            }
            else
            {
                label.style.backgroundColor = bgreset;
                if(regmon.test(checkbox)) { document.getElementById('cb_monday').checked = false;
            document.getElementById('lb_monday').style.backgroundColor = "#00FF00";}
                if(regtue.test(checkbox)) { document.getElementById('cb_tuesday').checked = false;
            document.getElementById('lb_tuesday').style.backgroundColor = "#00FF00";}
                if(regwed.test(checkbox)) {document.getElementById('cb_wednesday').checked = false;
            document.getElementById('lb_wednesday').style.backgroundColor = "#00FF00"; }
                if(regthu.test(checkbox)) { document.getElementById('cb_thursday').checked = false;
            document.getElementById('lb_thursday').style.backgroundColor = "#00FF00";}
                if(regfri.test(checkbox)) { document.getElementById('cb_friday').checked = false;
            document.getElementById('lb_friday').style.backgroundColor = "#00FF00";}
                if(regsat.test(checkbox)) { document.getElementById('cb_saturday').checked = false;
            document.getElementById('lb_saturday').style.backgroundColor = "#00FF00";}
                if(regsun.test(checkbox)) { document.getElementById('cb_sunday').checked = false;
            document.getElementById('lb_sunday').style.backgroundColor = "#00FF00";}
            }
        }
    }
}
```

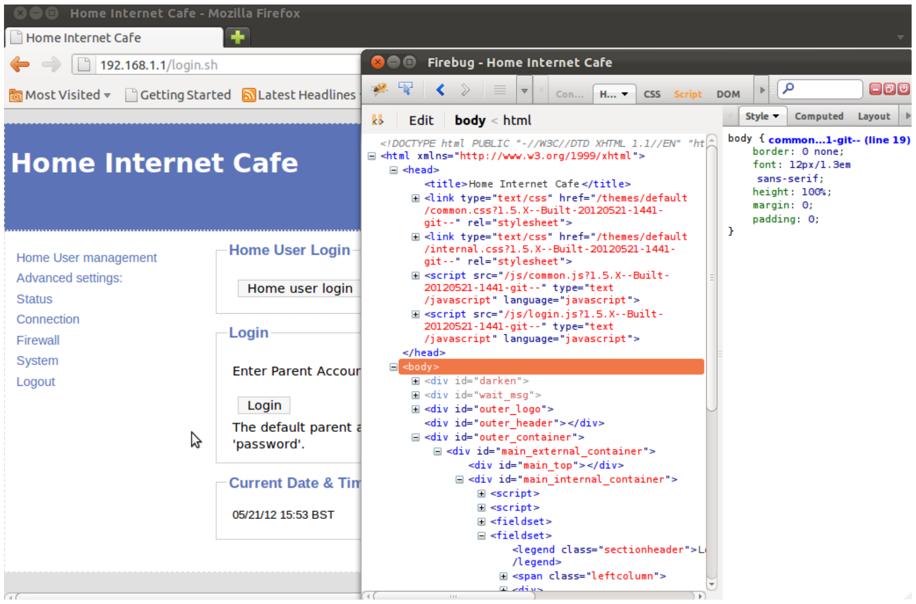
Figure 4.14: Debug using the error alert of Javascript lint

4.4.2 Firebug

With syntax errors fixed. We will build up our firmware and update it into our router. Then we have our implemented web interface. Newly developed web interface pages may contain lots of bugs. These bugs are related to shell scripts, html code, css files and javascript code. It is too complex to check each feature separately when meet errors. We use a famous debug tool Firebug [17]. Firebug is a web development tool based on the Firefox browser. It provides the functions include html and modify style editor, JavaScript debugger and network usage analysis. It is able for us to check all variables related to different programming language in the system together in this one tool.

Html/Css debug in Firebug

To access our web interface using Firefox browser, we access the shell script via connect to http://192.168.1.1/ from the firmware we built. In default setting system will redirect us to http://192.168.1.1/login.sh and lead us to different pages written as shell script with html code based on our operation then. When each interface web page is successfully loaded, we can enable Firebug via clicking the button in the panel or pressing F12 on our keyboard. When Firebug is enabled, it provide instant HTML and CSS views. Figure 4.15 shows the debug interface of Firebug when we loaded login page in home Internet cafe system. We can check the html and css code errors and directly change the code via Firebug. Note that the changes made only gives us the view of fixed result in the Firefox browser. The code inside router firmware is not changed. To fix html/css bugs we still need to go back to source code, have the bugs fixed, rebuild the firmware image and update image into the router. But we can find more bugs once via fixing some bugs directly when view the page to make new bugs appear.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "ht
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Home Internet Cafe</title>
    <link type="text/css" href="/themes/default
    /common.css?1.5.X--Built-20120521-1441-
    git--" rel="stylesheet">
    <link type="text/css" href="/themes/default
    /internal.css?1.5.X--Built-20120521-1441-
    git--" rel="stylesheet">
    <script src="/js/common.js?1.5.X--Built-
    20120521-1441-git--" type="text
    /javascript" language="javascript">
    <script src="/js/login.js?1.5.X--Built-
    20120521-1441-git--" type="text
    /javascript" language="javascript">
  </head>
  <body>
    <div id="darken">
    <div id="wait_msg">
    <div id="outer_logo">
    <div id="outer_header"></div>
    <div id="outer_container">
      <div id="main_external_container">
        <div id="main_top"></div>
        <div id="main_internal_container">
          <script>
          <script>
          <fieldset>
          <fieldset>
            <legend class="sectionheader">L
            /legend>
            <span class="leftcolumn">
            <div>
```

```
body { common...1-git-- (line 19)
    border: 0 none;
    font: 12px/1.3em
     sans-serif;
    height: 100%;
    margin: 0;
    padding: 0;
}
```

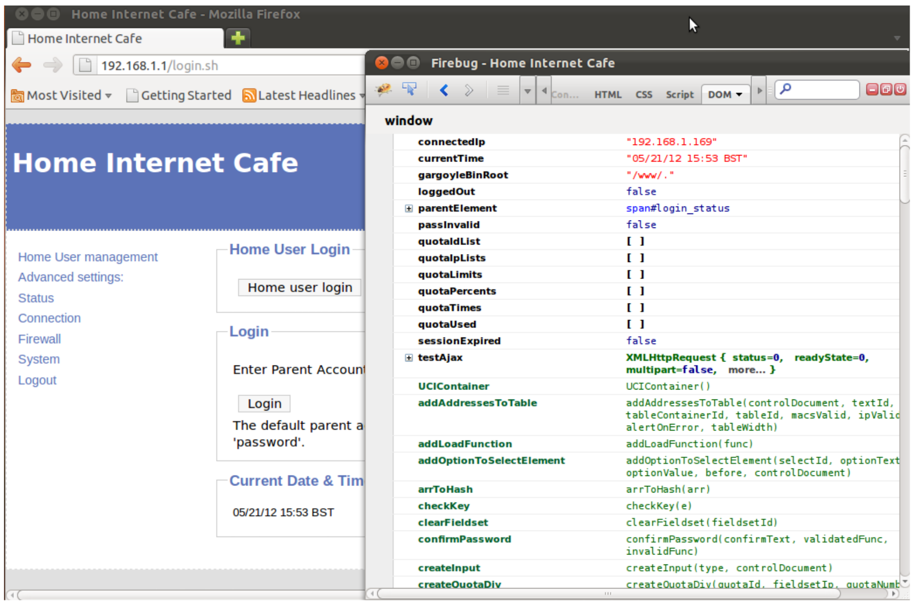
Figure 4.15: Html/css debug of our login page using Firebug

C/shell debug with Document object model

When we enable Firebug, another important feature is activated immediately named as Document Object Model(DOM). It is a great big hierarchy of objects and functions waiting to be tickled by Javascript. We mentioned in Subsection 4.2.4 talking about the code relationship that shell scripts get data from the system and have it operated by functions written in C code before being used by Javascript functions called by user actions. So the values and variables shown in document object model are those gained by shell scripts and C code. Then we can verify those values to find bugs contained in C code, although it is not displayed in Firebug interface. Figure 4.16 shows the document object model containing values and functions of login page in our system.

Javascript debug in Firebug

We should enable the Javascript debugger by ourselves after loading the page and open Firebug. Then we can choose different Javascript scripts based on user actions and related functions. We can add breakpoints easily by click target lines of the code. Also we can set target breakpoint pausing under specific conditions to avoid breaking too often. When a error occur, we can directly set a breakpoint at the line of the code where error occurred. We always have variables and functions watcher and can go step by step to check how target function works. Figure 4.17 shows the Javascript debugger working on the login page in our system.



Home Internet Cafe

Home User management
Advanced settings:
Status
Connection
Firewall
System
Logout

Home User Login

Home user login

Login

Enter Parent Accoun

Login

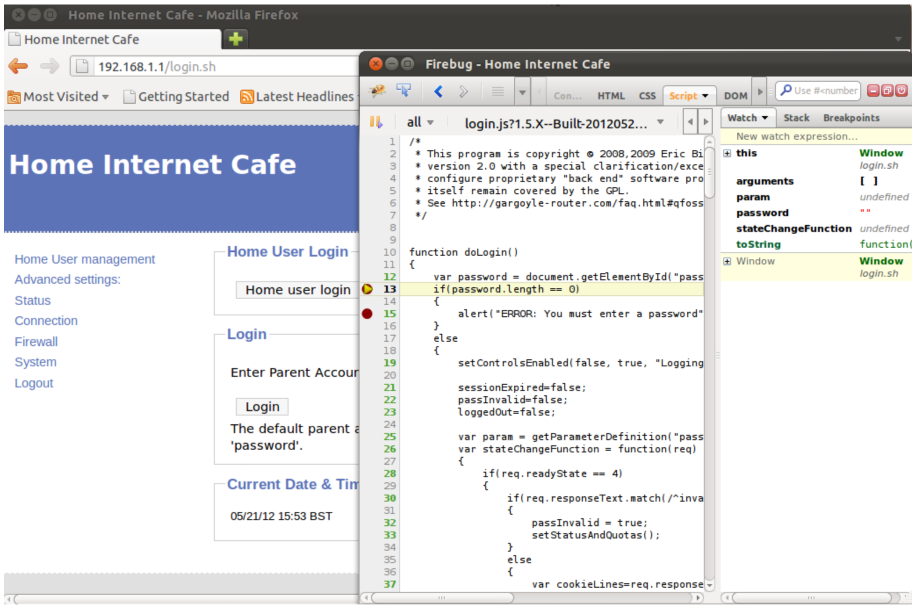
The default parent a
'password'.

Current Date & Tim

05/21/12 15:53 BST

```
window
connectedIp                 "192.168.1.169"
currentTime                 "05/21/12 15:53 BST"
gargoyleBinRoot             "/www/."
loggedOut                   false
⊞ parentElement             span#login_status
passInvalid                 false
quotaIdList                 [ ]
quotaIpLists                [ ]
quotaLimits                 [ ]
quotaPercents               [ ]
quotaTimes                  [ ]
quotaUsed                   [ ]
sessionExpired              false
⊞ testAjax                  XMLHttpRequest { status=0,  readyState=0,
                            multipart=false,  more... }
UCIContainer                UCIContainer()
addAddressesToTable         addAddressesToTable(controlDocument, textId,
                            tableContainerId, tableId, macsValid, ipValic
                            alertOnError, tableWidth)
addLoadFunction             addLoadFunction(func)
addOptionToSelectElement    addOptionToSelectElement(selectId, optionText
                            optionValue, before, controlDocument)
arrToHash                   arrToHash(arr)
checkKey                    checkKey(e)
clearFieldset               clearFieldset(fieldsetId)
confirmPassword             confirmPassword(confirmText, validatedFunc,
                            invalidFunc)
createInput                 createInput(type, controlDocument)
createQuotaDiv              createQuotaDiv(quotaId, fieldsetIp, quotaNumb
```

Figure 4.16: Document object model debug of our login page using Firebug



Home Internet Cafe

Home User management
Advanced settings:
Status
Connection
Firewall
System
Logout

Home User Login

Home user login

Login

Enter Parent Accour

Login

The default parent a
'password'.

Current Date & Tim

05/21/12 15:53 BST

```
/*
 * This program is copyright © 2008,2009 Eric Bi
 * version 2.0 with a special clarification/exce
 * configure proprietary "back end" software pro
 * itself remain covered by the GPL.
 * See http://gargoyle-router.com/faq.html#qfoss
 */


function doLogin()
{
    var password = document.getElementById("pass
    if(password.length == 0)
    {
        alert("ERROR: You must enter a password"
    }
    else
    {
        setControlsEnabled(false, true, "Logging

        sessionExpired=false;
        passInvalid=false;
        loggedOut=false;

        var param = getParameterDefinition("pass
        var stateChangeFunction = function(req)
        {
            if(req.readyState == 4)
            {
                if(req.responseText.match(/^inva
                {
                    passInvalid = true;
                    setStatusAndQuotas();
                }
                else
                {
                    var cookieLines=req.response
```

```
⊞ this                  Window
                        login.sh
  arguments             [ ]
  param                 undefined
  password              ""
  stateChangeFunction   undefined
  toString              function(
⊞ Window                Window
                        login.sh
```

Figure 4.17: Javascript debug debug of our login page using Firebug

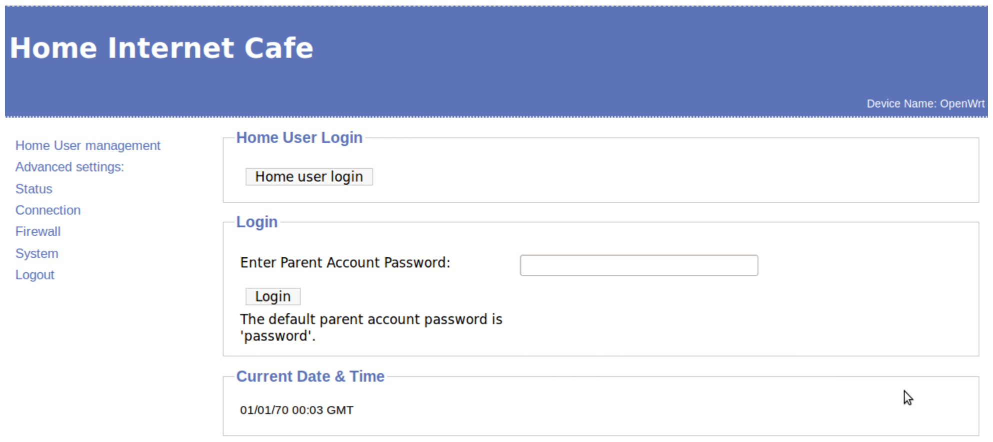
Chapter 5

Result and Testing

As we are developing a system server for home user and focus on providing functions required by user and interface friendly to user. Test work and user feedback is very important to us before, during and after system design and implementation. The important test work starts when we use paper prototype to design the interface and will continue even after we finish the system development - as we always need to improve developed system with user feedback and update it with fixing bugs found. In this chapter, we will first show the key web interface pages of our system to show the implementation result. Then we will introduce our user test works and system evaluate.

5.1 User Interface overview

After the router boot. Via access the address 192.168.1.1, our user will get into our home Internet cafe system interface and see the welcome page as shown in Figure 5.1.



Enter Parent Account Password:

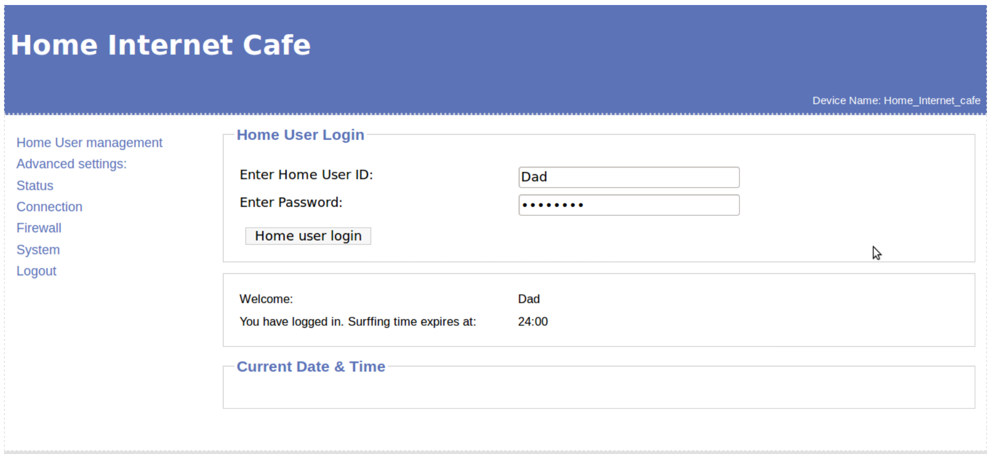
The default parent account password is 'password'.

01/01/70 00:03 GMT

Figure 5.1: Welcome page of home Internet cafe

Home user Login

Parent account login by typing their password in the parent login area of the welcome page. Home users who want Internet access should click home user login button and move to home user login page. When they type right home user code, they can login as home user and start using Internet based on their home user restriction settings and get the login information shown in Figure 5.2.

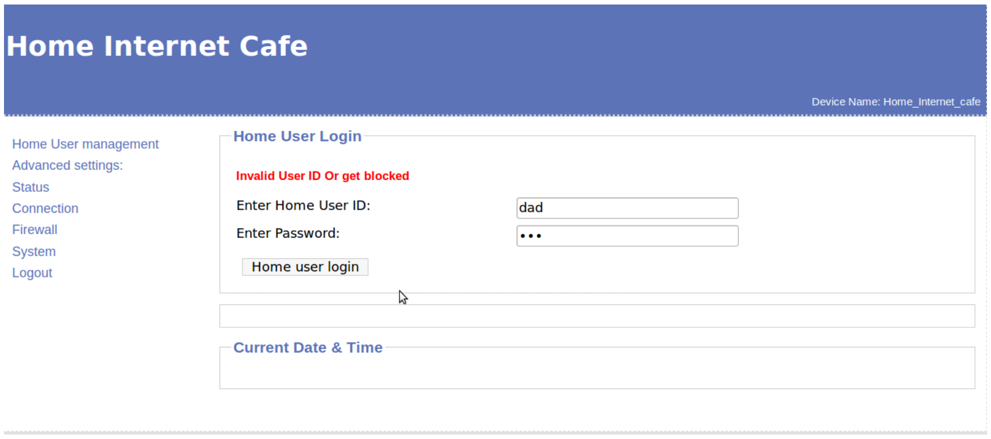


Enter Home User ID: Dad
Enter Password: ••••••••

Welcome: Dad
You have logged in. Surffing time expires at: 24:00

Figure 5.2: Home user login page

When users type wrong username or password or the target user they want to login as is blocked, we have notification for them as shown in Figure5.3.

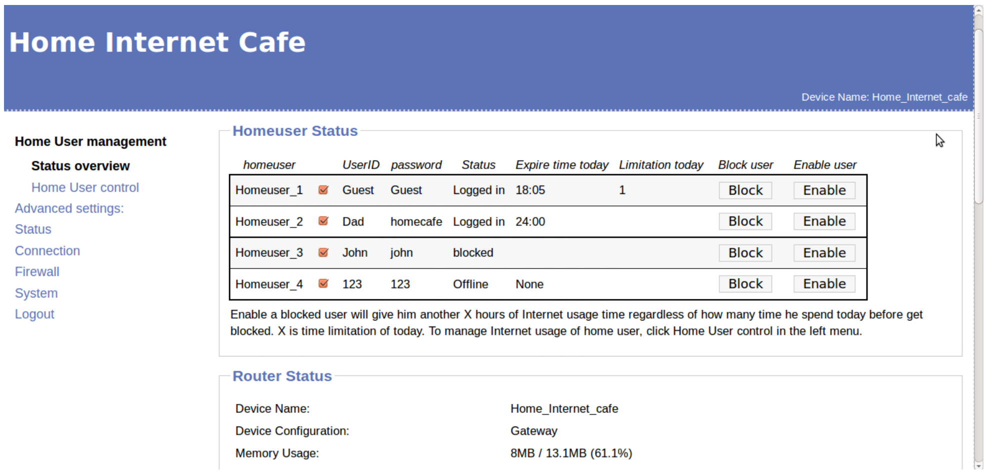


Invalid User ID Or get blocked
Enter Home User ID: dad
Enter Password: •••

Figure 5.3: Notification for wrong home user or get blocked

Home user management

Parent account have the ability to handle settings of home users and manage their Internet. When user logged in as parent account they will be redirected to home user management page as shown in Figure 5.4.



Homeuser Status

homeuser		UserID	password	Status	Expire time today	Limitation today	Block user	Enable user
Homeuser_1	✓	Guest	Guest	Logged in	18:05	1	Block	Enable
Homeuser_2	✓	Dad	homecafe	Logged in	24:00		Block	Enable
Homeuser_3	✓	John	john	blocked			Block	Enable
Homeuser_4	✓	123	123	Offline	None		Block	Enable

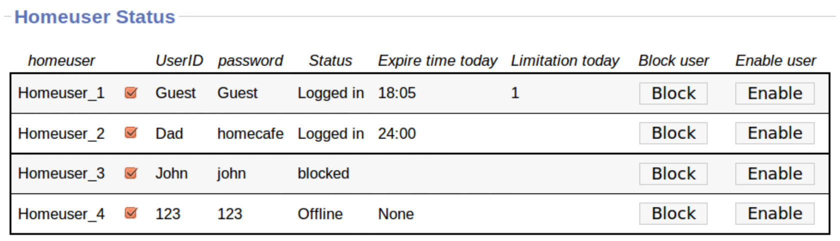
Enable a blocked user will give him another X hours of Internet usage time regardless of how many time he spend today before get blocked. X is time limitation of today. To manage Internet usage of home user, click Home User control in the left menu.

Router Status

Device Name:	Home_Internet_cafe
Device Configuration:	Gateway
Memory Usage:	8MB / 13.1MB (61.1%)

Figure 5.4: Home user management interface page

In default the redirected page shows the status of router and home users. We have a table for home users showing their status, as shown with details in Figure **??**


Homeuser Status

homeuser		UserID	password	Status	Expire time today	Limitation today	Block user	Enable user
Homeuser_1	✓	Guest	Guest	Logged in	18:05	1	Block	Enable
Homeuser_2	✓	Dad	homecafe	Logged in	24:00		Block	Enable
Homeuser_3	✓	John	john	blocked			Block	Enable
Homeuser_4	✓	123	123	Offline	None		Block	Enable

Figure 5.5: Home user status table

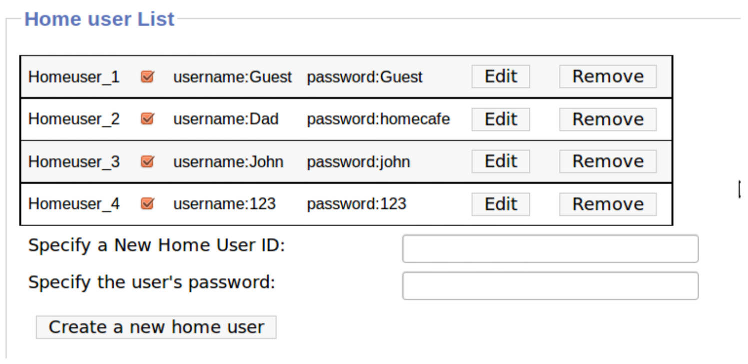
In the table we can find different user status. It shows home users are logged in, offline or get blocked. And parent account can see the expire time of target logged in user based on the time restriction settings, which can tell us how much Internet usage time left for target user when comparing the expire time with current time. When click 'block user' or 'enable' we can set user status between blocked or enabled to directly reject target user in special cases.

It is necessary to provide functions to edit and create user, User can access user edit page from the left menu to choose the Home user control option and get to the home user control panel. The interface of home user creator is shown in details in Figure 5.6.

In the interface we can see the home user list table. By click the edit button we can access the panel of home user editor to reset the home user's ID and password as shown in Figure 5.7.

Time restriction

We have introduced the time restriction settigns in Section 4.3.2. By click the edit button in the home user list table we can access the time restriction settings below the home user edit panel. Part of the detailed restriction settings is shown in Figure 5.8.



Home user List

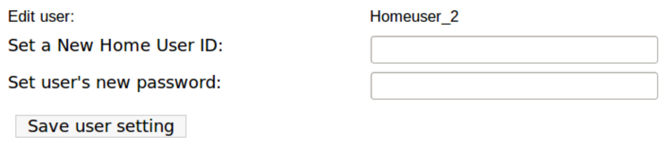
Homeuser_1	✓	username:Guest	password:Guest	Edit	Remove
Homeuser_2	✓	username:Dad	password:homecafe	Edit	Remove
Homeuser_3	✓	username:John	password:john	Edit	Remove
Homeuser_4	✓	username:123	password:123	Edit	Remove

Specify a New Home User ID:

Specify the user's password:

Create a new home user

Figure 5.6: Parent interface to edit and create home user



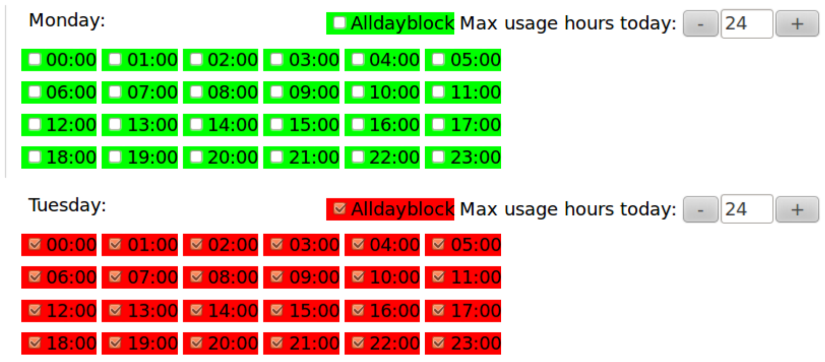
Edit user: Homeuser_2

Set a New Home User ID:

Set user's new password:

Save user setting

Figure 5.7: Parent interface to edit and create home user



Monday: ☐ Alldayblock Max usage hours today: - 24 +

☐ 00:00	☐ 01:00	☐ 02:00	☐ 03:00	☐ 04:00	☐ 05:00
☐ 06:00	☐ 07:00	☐ 08:00	☐ 09:00	☐ 10:00	☐ 11:00
☐ 12:00	☐ 13:00	☐ 14:00	☐ 15:00	☐ 16:00	☐ 17:00
☐ 18:00	☐ 19:00	☐ 20:00	☐ 21:00	☐ 22:00	☐ 23:00

Tuesday: ✓ Alldayblock Max usage hours today: - 24 +

✓ 00:00	✓ 01:00	✓ 02:00	✓ 03:00	✓ 04:00	✓ 05:00
✓ 06:00	✓ 07:00	✓ 08:00	✓ 09:00	✓ 10:00	✓ 11:00
✓ 12:00	✓ 13:00	✓ 14:00	✓ 15:00	✓ 16:00	✓ 17:00
✓ 18:00	✓ 19:00	✓ 20:00	✓ 21:00	✓ 22:00	✓ 23:00

Figure 5.8: Home user time restriction settings

Everyday is divided into 24 time period in our settings displayed as 24 checkboxes. For example, the checkbox with the name ’00:00’ means the time period from 00:00-01:00 at current day. Green checkbox means user is allowed to access the Internet at target time period, red checkbox means user will get blocked during the time period. And we have a total usage time limitation of each day. Where user can increase or decrease number of hours for total Internet use by the ’-’ and ’+’ buttons. They can also type the number by themselves, but the final number is forced as a integer between 0 and 24. By click ’Alldayblock’ parent can deny target user’s Internet access for the whole day. Whenever a time period is activated into green, ’Alldayblock’ checkbox will turn to unchecked status and shown as green then.

5.2 Performance evaluate

The usability of our system is provided and improved based on users' feedback from the user testing principles talked in Section 3.3. Another important factor of our system is the efficiency. We did some performance evaluations to verify the efficiency of our system. All these evaluations are done with different router firmwares in the Linksys Wireless-G broadband router WRT54GL v1.1 as we used for implementation. To compare the efficiency of different router firmwares, we choose the official firmware provided by Linksystem, the default OpenWrt firmware and our home Internet cafe system developed based on OpenWrt.

5.2.1 Router booting time

It always take more time for router to boot when using third party firmware. It is because third party firmware is designed for a set of different routers instead of the specific router. Third party firmware have some more processes when booting to fit target router's hardware with compatible drivers. This will cost more calculation and lead to longer delay. Also we have some pre-settings which will be applied to system during the booting task. This pre-settings applying task will also slow down the booting process. User always don't want to wait for too long time when they need Internet immediately. So the booting time can be very important in user experience. The booting time comparison is shown in table 5.1.

	Linksystem	**OpenWrt**	**Home Internet cafe**
booting time	8 s	40 s	83 s

Table 5.1: Booting time of different router firmwares

5.2.2 User settings operation time

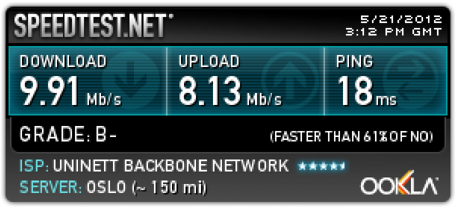
As most of our designed functions are related to firewall operations, it will increase the operation time when system execute commands which will restart the firewall. And the work of activating the wireless access will both restart the firewall and apply different settings. This makes the process longer. Our system is very weak in saving time of firewall related functions because we have more settings applied and commands executed during the operation. The default firmware can finish the task really fast because it is specially designed for target hardware. While third party firmwares always need to do some mapping between high layer command and low layer command with related functions. The user settings operation time comparison is shown in table 5.3.

	Linksystem	OpenWrt	Home Internet cafe
Firewall restart operation	4 s	22 s	37 s
Activate wireless access	1 s	5 s	108 s

Table 5.2: User settings operation time of different router firmwares

5.2.3 Download/Upload speed

One of the most important user experience features is the Internet speed. Since our system is using iptables with a long list of restriction rules, whether it will decrease the Internet speed or not will be very important. To test the Internet speed, we use the online Internet speed testing web site 'http://www.speedtest.net/'. It will provide ping time and both download and upload speed of our current network in use and give out diagram as shown in Figure 5.9.



SPEEDTEST.NET 5/21/2012 3:12 PM GMT

DOWNLOAD 9.91 Mb/s UPLOAD 8.13 Mb/s PING 18 ms

GRADE: B- (FASTER THAN 61% OF NO)

ISP: UNINETT BACKBONE NETWORK ★★★★★

SERVER: OSLO (~ 150 mi)

OOKLA

Figure 5.9: Result diagram of the Internet speed test

With the help of speed test web site, we got the Internet speed test of three router firmwares and shown the results in table 5.3.

	Linksystem	OpenWrt	Home Internet cafe
Ping time	20 ms	20 ms	18 ms
Download speed	9.40 Mb/s	9.91 Mb/s	9.70 Mb/s
Upload speed	6.19 Mb/s	8.13 Mb/s	7.85 Mb/s

Table 5.3: User settings operation time of different router firmwares

Some random features exist so the 2ms less ping time can't stand for anything that show our system has better performance. As third party firmwares have better packets handling method and improved functions, they for sure will improve the router's performance as shown in download and upload speed. But our system have some decreasing data comparing with normal OpenWrt firmware. That's the fault of long iptables rule lists, which are used for providing new time restriction features.

Chapter 6

Conclusion and further work

6.1 Conclusion

Our goal is to design a home Internet cafe as a user friendly Internet access control system for family users. To achieve this goal, we focus on both the user friendly interface design work and the router Internet usage control functions development work. In the interface design work part, we introduce a design life cycle from paper prototype design to mock-up web interface sample and finally reach the real implementation for the system. During these different steps, we test our interface with different users and improve it based on user feedback. 6 users are involved during paper prototype testing. They provide in total 12 user tests as some of them test our interface more than twice. We involve 5 users to test and give feedback on the mock-up prototype for 8 times. And for the implemented final system, we involve 10 users in the testing work to get their feedback. During the test, the parents that we interviewed all think the concept of home Internet cafe is an excellent concept. They are very worried about their children who have played on the Internet too much. And they mentioned they have no efficient way to control the Internet usage of their children. So our system is welcome to them.

In the functions design part, we go step by step with lots of tasks to make our development of the firmware go deeper and deeper. Start from doing research on the state of the art of different parents control systems using different solutions, we pick out the best one fits our demand. After that we choose a suitable firmware to build our system based on it. We get a deep understanding of the firmware via checking out and compiling the source code, reading the related code written in different programming language and developing some additional features based on the existing code structures and functions. Then we manage to develop separated functions and attach them into the router firmware to turn it into the home Internet cafe system with advanced functions for family users. We extend the packet filter rule set with own rules to provide better parental control functionalities. And during the development we implement our own interface designed and tested by users to operate the newly developed system. Our two focused tasks are combined together into the final Internet access control system then to achieve our goal as providing user friendly home Internet access control system with advanced functionalities.

6.2 Discussion and contributions

We have shown the principles of our system functions and how its interface looks like in chapter 4 and chapter 5, and provided some performance evaluation results comparing with normal router firmware. With the basic hardware functions the same, our final system designed looks and works very differently from the Openwrt firmware we chosen in the functions for user control. We change current functions and build new functions to realize new functionalities. During these works, new features are provided but the router has to execute more complex commands and do more calculations. We sacrifice some level of the system performance to exchange for advanced functionalities in this way. We will introduce our system advantages as our contribution and have some discussion about the lost system performance as the disadvantages below.

6.2.1 System advantages

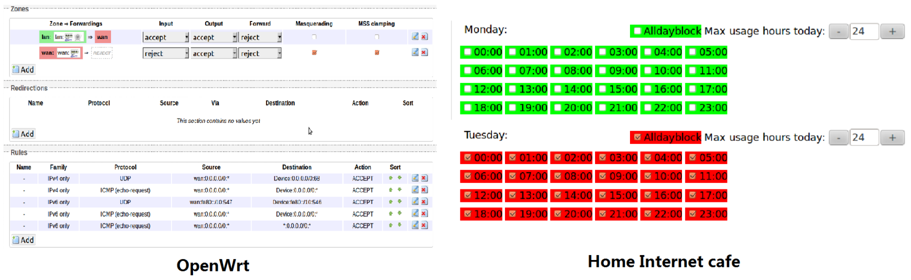
New features

We provide two key new features in our system based on the requirement of parents: code identify system and total Internet usage time limitation. With the name of ’home Internet cafe’, we want to provide our users a convenient management method similar to the Internet cafe. So we designed our system to provide Internet access to home users based on home user ID, so called code identify system. This control method not only makes users’ management works more convenient and clear but also provides better usability for users, which we will talk later.

It is a strange thing that we can never find access restriction control panels of different router firmwares that have the function providing total Internet usage time limitation. From the developer’s view point, this function may be useless as we have another way to realize similar limitation by setting up exact restriction time period with limited hours. But total limitation hours can be very useful to parents in different cases when they don’t want to set up exact time for each weekday and change the settings every week. Our functionalities providing total Internet usage time limitation combined with blocked time period restriction system can be very helpful to parent users then.

Better usability

The most important feature in our system is home Internet usage management. The functions and interface we designed all serve for this task. As we want to provide user friendly interface to our users, we keep testing and improving the interface to gain better usability. Normal third party firmwares are really well designed with many advanced settings and improved performance. But at most time the features are too complex for normal home users with less Internet skills to understand. Figure 6.1 compares Openwrt time restriction settings panel and our interface for time limitation.



Zones

Zone ⇒ Forwardings	Input	Output	Forward	Masquerading	MSS clamping
lan: lan: ⇒ wan	accept	accept	reject	☐	☐
wan: wan: → REJECT	reject	accept	reject	☑	☑

Add

Redirections

Name	Protocol	Source	Via	Destination	Action	Sort

This section contains no values yet

Add

Rules

Name	Family	Protocol	Source	Destination	Action	Sort
-	IPv4 only	UDP	wan:0.0.0.0/0:*	Device:0.0.0.0/0:68	ACCEPT	
-	IPv4 only	ICMP (echo-request)	wan:0.0.0.0/0:*	Device:0.0.0.0/0:*	ACCEPT	
-	IPv6 only	UDP	wan:fe80::/10:547	Device:fe80::/10:546	ACCEPT	
-	IPv6 only	ICMP (echo-request)	wan:0.0.0.0/0:*	Device:0.0.0.0/0:*	ACCEPT	
-	IPv6 only	ICMP (echo-request)	wan:0.0.0.0/0:*	*:0.0.0.0/0:*	ACCEPT	

Add

OpenWrt

Monday: ☐ Alldayblock Max usage hours today: - 24 +

☐ 00:00	☐ 01:00	☐ 02:00	☐ 03:00	☐ 04:00	☐ 05:00
☐ 06:00	☐ 07:00	☐ 08:00	☐ 09:00	☐ 10:00	☐ 11:00
☐ 12:00	☐ 13:00	☐ 14:00	☐ 15:00	☐ 16:00	☐ 17:00
☐ 18:00	☐ 19:00	☐ 20:00	☐ 21:00	☐ 22:00	☐ 23:00

Tuesday: ☑ Alldayblock Max usage hours today: - 24 +

☑ 00:00	☑ 01:00	☑ 02:00	☑ 03:00	☑ 04:00	☑ 05:00
☑ 06:00	☑ 07:00	☑ 08:00	☑ 09:00	☑ 10:00	☑ 11:00
☑ 12:00	☑ 13:00	☑ 14:00	☑ 15:00	☑ 16:00	☑ 17:00
☑ 18:00	☑ 19:00	☑ 20:00	☑ 21:00	☑ 22:00	☑ 23:00

Home Internet cafe

Figure 6.1: Comparison of Openwrt and Home Internet cafe time restriction interface

Many low level users won't understand what are the zones, redirections talking about and may not know the protocol or even source/destination IP addresses in the Openwrt interface. But in our system with the code identify system, users can easily manage users' usage and set up rules based on different user ID. They don't even need to understand IP address and MAC address as these features will be handled automatically by our system. Also the block view of time restriction can be very convenient to operate with green and red colors stand for active/block status since it provides a graphical view for our users.

6.2.2 System disadvantages

Some evaluation results our system are shown in section 5.2. Although the Internet speeds don't get limited too much. We can find that our system reacts really slowly to user operations related to firewall. Users always hate long waiting time in modern high speed daily life. However, to realize advanced features with less obstructions for user operation, we have lots of functions and command executed during background processing. The booting time and enabling time of wireless access take extremely long time because of this. A tolerable thing is that users won't reboot their routers or switch on/off wireless access too frequently. But our system does operations related to firewall slowly also. For example, our users need to wait for 30+ seconds when setting up time limitations for a home user. This long waiting time will lower our system's user experience a lot.

6.3 Further work

6.3.1 Advanced useful functions

As we finished the basic features to provide home Internet cafe system for parent users, there are several features which we don't have time to make them realized. One feature is the idea of providing a timer for target user telling them left Internet usage time and allowing them to pause the timer and get offline status to save their usage time. Currently we only inform users when will their Internet usage time get expired. It will be more humanistic to provide this timer stop function. But we should keep the functions from abused. Users may stop the timer every time they have successfully loaded a page. It's a cheat which will make the total time limitation useless. A limitation of total times of stop actions will help to avoid this cheat.

Another feature is the web page redirection. In our system user need to log in via access the router interface from 192.168.1.1. And they will get connection error notification from their browsers when try to visit other web sites without information telling them to login as home user. It will be very nice if our users can be redirect to the home user login page when they try to visit other web sites, just like what we usually meet in the airport wireless services or wifi hot spot.

6.3.2 Continuing interface improvement

With our design and improvement, our user interface is more friendly and convenient comparing with normal third party firmwares developed for advanced users with more Internet skills. But there still may be some features missing and detailed guidelines needed for our users since we will never know what our users will do to the system and how they can understand the options with the right meanings. It is really necessary to continue the interface improvement works with more user testing and feedback.

6.3.3 Better system performance

We discussed our system and find the biggest disadvantage is the long reaction time. To have better user experience we need to improve our system with better performance in the speed of task processing. We should go deep into the system structure and separated steps of system operation to find out the cause of the long processing time. By fixing problems or improving algorithms we may be able to reduce the reaction times to provide better system performance.

Bibliography

[1] P. Anne. ”surfbalanc kid-safe browser app now available for android users”. PR Works, Inc. [Online]. Available: http://www.surfbalance.com/SurfBalanceAndroid_v1_0.pdf

[2] ”apache subversion features”. Apache Software Foundation. [Online]. Available: http://subversion.apache.org/features.html

[3] ”apache tutorial: Dynamic content with cgi”. Apache Software Foundation. [Online]. Available: http://httpd.apache.org/docs/2.0/howto/cgi.html

[4] ”optimized 802.11g router with broadrange”. Broadcom corporation. [Online]. Available: http://www.broadcom.com/collateral/pb/5352EL-PB00-R.pdf

[5] ”about buddy browser”. Buddy Browser. [Online]. Available: http://www.buddybrowser.com/about-us.html

[6] ”wireless-g broadband router(wrt54gl)”. CISCO, Inc. [Online]. Available: http://homesupport.cisco.com/en-eu/support/routers/WRT54GL

[7] ”net nanny parental controls overview”. ContentWatch, Inc. [Online]. Available: http://www.netnanny.com/features

[8] ”a setup simulation of v24beta-dd-wrt interface”. DDWRT project. [Online]. Available: http://www.informatione.gmxhome.de/DDWRT/Standard/V24BetaVPN/Filters.html

[9] ”what is dd-wrt?”. DDWRT project. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/What_is_DD-WRT%3F

[10] ”my mobile watchdog is loaded with new features and is easier to use”. eAgency, Inc. [Online]. Available: http://www.mymobilewatchdog.com/productinfo.shtml

[11] B. Eric. ”developer documentation”. gargoyle project. [Online]. Available: http://www.gargoyle-router.com/wiki/doku.php?id=developer_documentation

[12] ”gcc, the gnu compiler collection”. Free Software Foundation, Inc. [Online]. Available: http://gcc.gnu.org/

[13] ”glibc, the gnu c library”. Free Software Foundation, Inc. [Online]. Available: http://www.gnu.org/software/libc/

[14] ”gnu binutils”. Free Software Foundation, Inc. [Online]. Available: http://www.gnu.org/software/binutils/

[15] ”what is safe eyes and how can it help my family?”. InternetSafety.coms, Inc. [Online]. Available: http://www.internetsafety.com/safe-eyes-parental-control-software.php

[16] ”kids have fun and parents rest easy”. KidZui.com. [Online]. Available: http://www.kidzui.com/learn_more

[17] ”what is firebug”. Mozilla Corporation. [Online]. Available: http://getfirebug.com/whatisfirebug

[18] ”haserl on sourceforge”. N. Angelacos. [Online]. Available: http://subversion.apache.org/features.html

[19] ”openwrt manual”. openwrt project. [Online]. Available: http://downloads.openwrt.org/kamikaze/docs/openwrt.html

[20] ”the uci system”. openwrt project. [Online]. Available: http://wiki.openwrt.org/doc/uci

[21] A. Oskar. ”iptables tutorial 1.2.2”. Netfilter Core Team. [Online]. Available: http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html#ABOUTTHEAUTHOR

[22] B. Saikat. ”pencil project turn firefox into a diagramming and prototyping tool”. Makeuserof, Website. [Online]. Available: http://www.makeuseof.com/tag/pencil-project-turn-firefox-diagramming-prototyping-tool/

[23] L. Sarah, “Private uses in public spaces : A study of an internet cafe,” *New Media Society*, vol. 1(3), pp. 331–350, 1999.

[24] ”protect children from unsuitable web content”. SOFTONIC INTERNATIONAL S.L. [Online]. Available: http://glubble_for_families.en.softonic.com/

[25] ”javascript lint”. SourceForge Project. [Online]. Available: http://www.javascriptlint.com/index.htm

[26] K. Steve, *Don’t Make Me Think: Common Sense Approach to Web Usability*. New Riders Publishing, 2000.

[27] ”a smarter way to keep kids safe online”. Symantec Corporation. [Online]. Available: https://onlinefamily.norton.com/familysafety/loginStart.fs?ULang=eng

[28] ”about ubuntu:the ubuntu story”. Ubuntu community. [Online]. Available: http://www.ubuntu.com/project/about-ubuntu

Appendices

Appendix A.

Source code

The source code of the interface and function part is updated in Google code. To get the source code, checkout it using svn via http:

```
svn checkout http://home-internet-cafe.googlecode.com/svn/trunk/
```

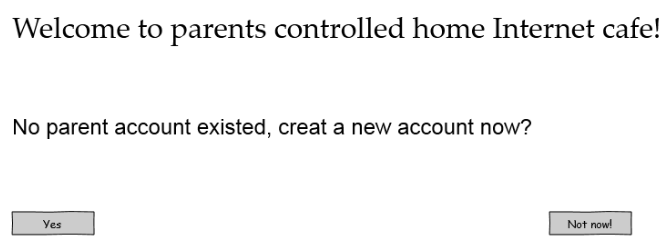
Appendix B.

Abbreviations

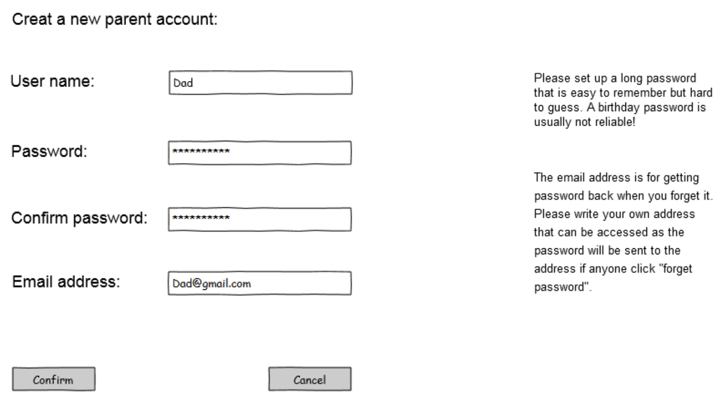
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	HyperText Markup Language
ISP	Internet Service Provider
MAC	Media Access Control
OEM	Original Equipment Manufacturer
OS	Operating System
P2P	Peer to Peer
TFTP	Trivial File Transfer Protocol
UCI	Unified Configuration Interface
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network

Appendix C.

Mock-up interface prototype


Welcome to parents controlled home Internet cafe!

No parent account existed, creat a new account now?


Creat a new parent account:

User name: Dad

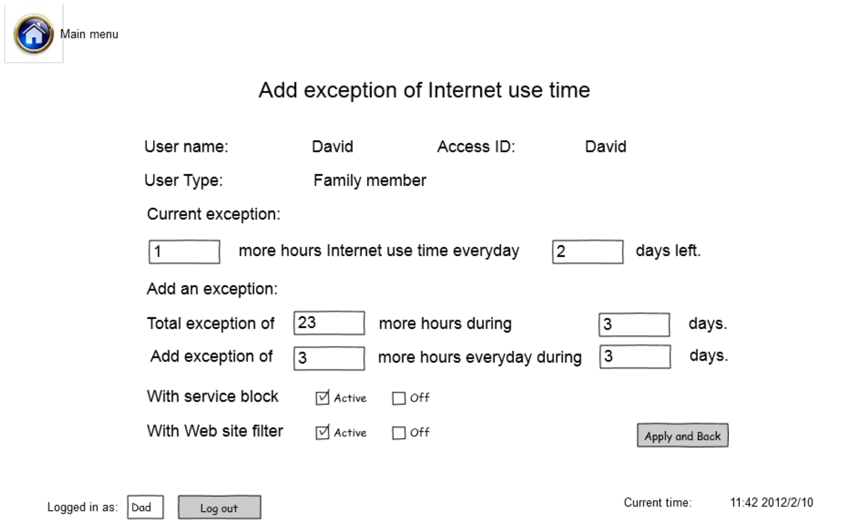
Password: **********

Confirm password: **********

Email address: Dad@gmail.com

Please set up a long password that is easy to remember but hard to guess. A birthday password is usually not reliable!

The email address is for getting password back when you forget it. Please write your own address that can be accessed as the password will be sent to the address if anyone click "forget password".



Main menu

Add exception of Internet use time

User name: David Access ID: David

User Type: Family member

Current exception:

1 more hours Internet use time everyday 2 days left.

Add an exception:

Total exception of 23 more hours during 3 days.

Add exception of 3 more hours everyday during 3 days.

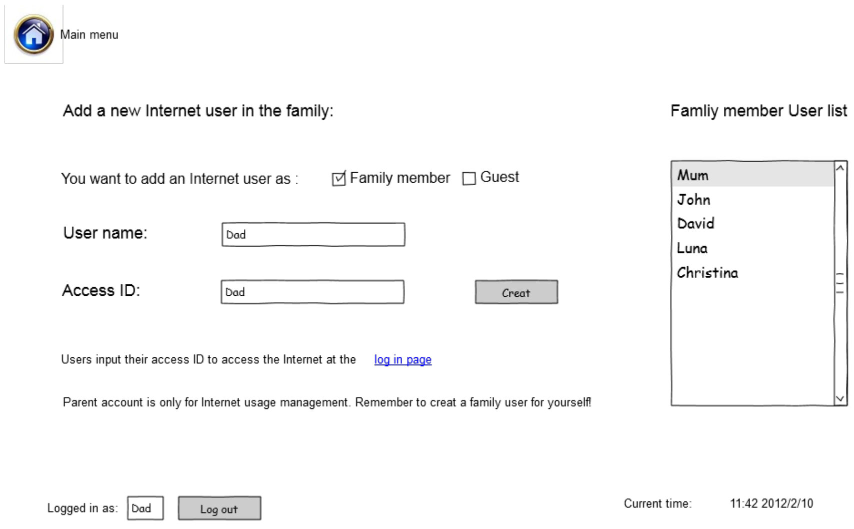
With service block [✓ Active] Off

With Web site filter [✓ Active] Off

Apply and Back

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Add a new Internet user in the family:

Famliy member User list

You want to add an Internet user as : [✓ Family member] Guest

User name: Dad

Access ID: Dad

Creat

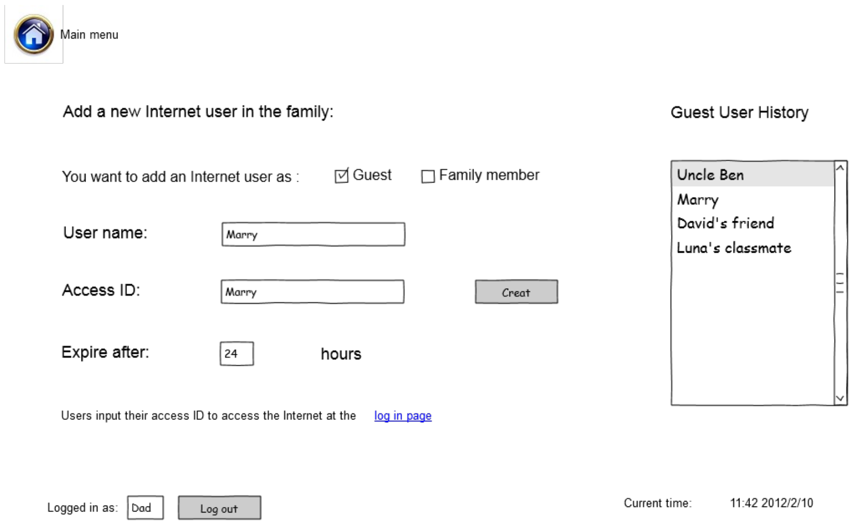
Mum
John
David
Luna
Christina

Users input their access ID to access the Internet at the log in page

Parent account is only for Internet usage management. Remember to creat a family user for yourself!

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Add a new Internet user in the family:

Guest User History

You want to add an Internet user as : [✓ Guest] Family member

User name: Marry

Access ID: Marry

Creat

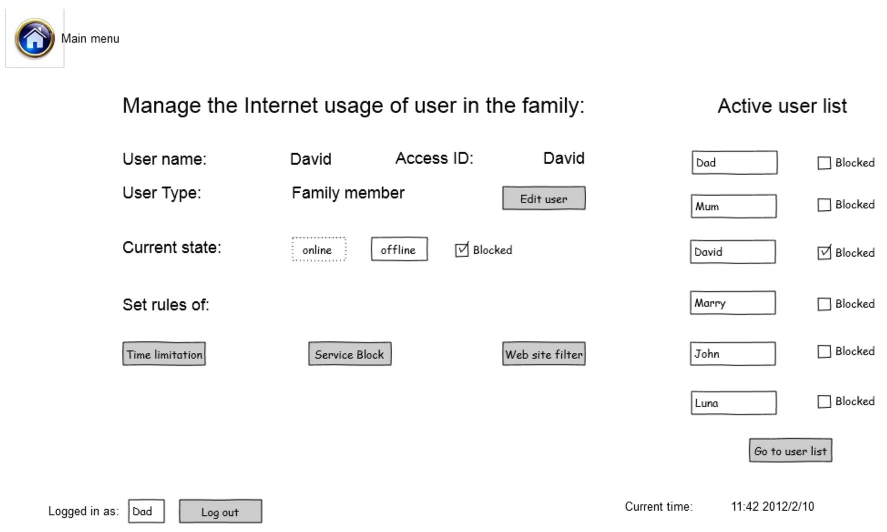
Expire after: 24 hours

Uncle Ben
Marry
David's friend
Luna's classmate

Users input their access ID to access the Internet at the log in page

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Manage the Internet usage of user in the family:

User name: David Access ID: David

User Type: Family member

Current state: online offline [✓ Blocked]

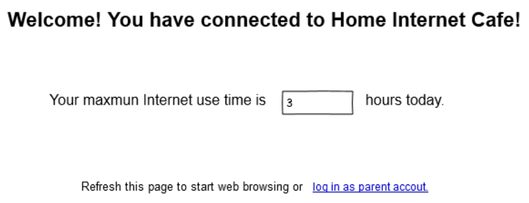
Set rules of:

Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☑ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Logged in as: Dad

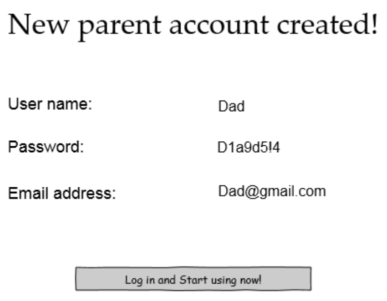
Current time: 11:42 2012/2/10



Welcome! You have connected to Home Internet Cafe!

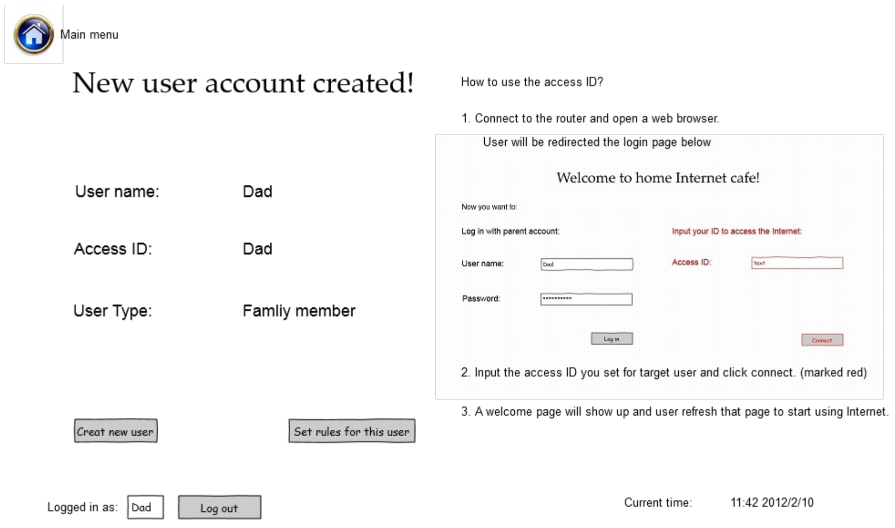
Your maxmun Internet use time is 3 hours today.

Refresh this page to start web browsing or log in as parent accout.



New parent account created!

User name:	Dad
Password:	D1a9d5!4
Email address:	Dad@gmail.com



Main menu

New user account created!

User name: Dad

Access ID: Dad

User Type: Famliy member

How to use the access ID?

1. Connect to the router and open a web browser.

User will be redirected the login page below

Welcome to home Internet cafe!

Now you want to:

Log in with parent account:

User name: Dad

Password: **********

Input your ID to access the Internet:

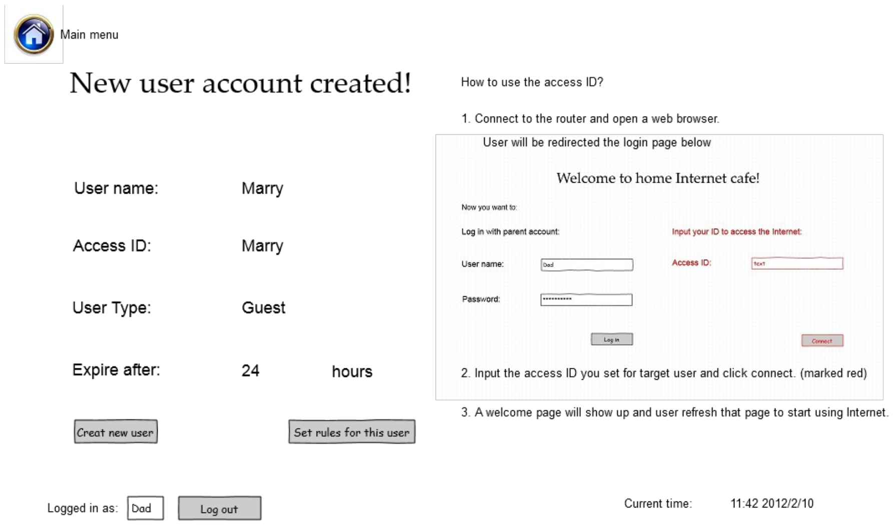
Access ID: text

2. Input the access ID you set for target user and click connect. (marked red)

3. A welcome page will show up and user refresh that page to start using Internet.

Logged in as: Dad

Current time: 11:42 2012/2/10



Main menu

New user account created!

User name: Marry

Access ID: Marry

User Type: Guest

Expire after: 24 hours

How to use the access ID?

1. Connect to the router and open a web browser.

User will be redirected the login page below

Welcome to home Internet cafe!

Now you want to:

Log in with parent account:

User name: Dad

Password: **********

Input your ID to access the Internet:

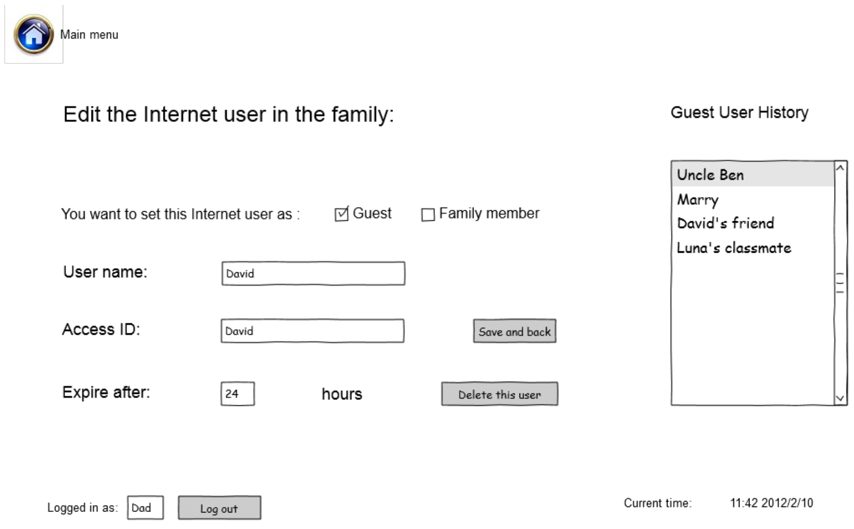
Access ID: text

2. Input the access ID you set for target user and click connect. (marked red)

3. A welcome page will show up and user refresh that page to start using Internet.

Logged in as: Dad

Current time: 11:42 2012/2/10



Main menu

Edit the Internet user in the family:

You want to set this Internet user as : [✓ Guest] Family member

User name: David

Access ID: David

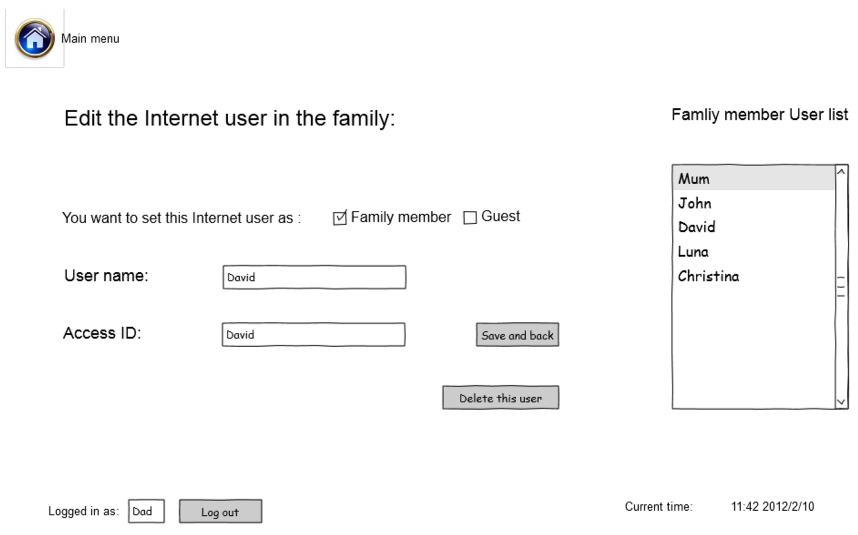
Expire after: 24 hours

Guest User History

- Uncle Ben
- Marry
- David's friend
- Luna's classmate

Logged in as: Dad

Current time: 11:42 2012/2/10



Edit the Internet user in the family:

You want to set this Internet user as : [✓ Family member] Guest

User name: David

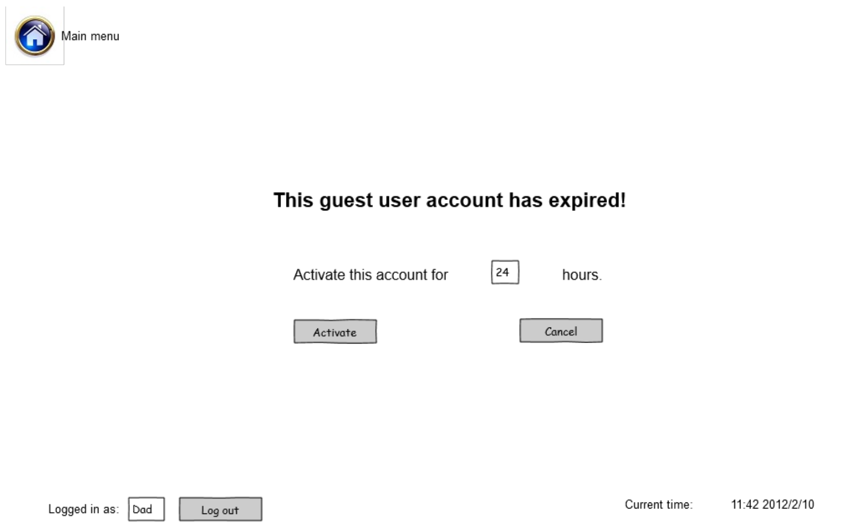
Access ID: David

Famliy member User list

- Mum
- John
- David
- Luna
- Christina

Logged in as: Dad

Current time: 11:42 2012/2/10

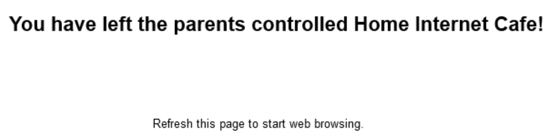


This guest user account has expired!

Activate this account for 24 hours.

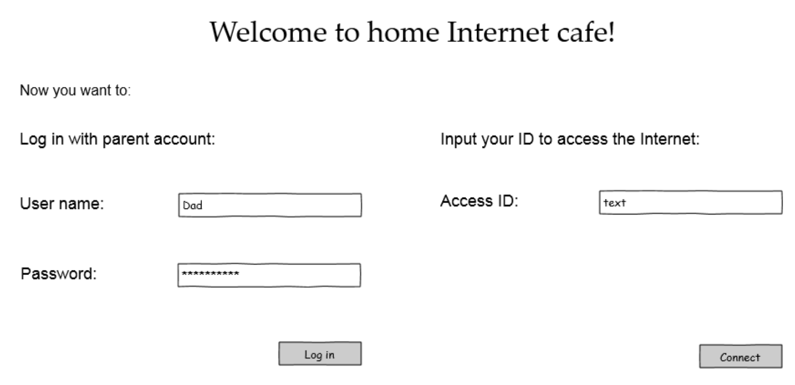
Logged in as: Dad

Current time: 11:42 2012/2/10



You have left the parents controlled Home Internet Cafe!

Refresh this page to start web browsing.



Welcome to home Internet cafe!

Now you want to:

Log in with parent account:

User name: Dad

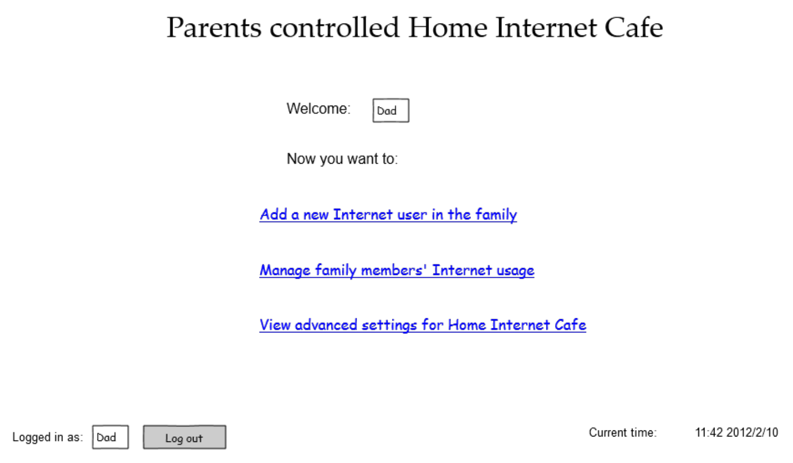
Password: **********

Log in

Input your ID to access the Internet:

Access ID: text

Connect



Parents controlled Home Internet Cafe

Welcome: Dad

Now you want to:

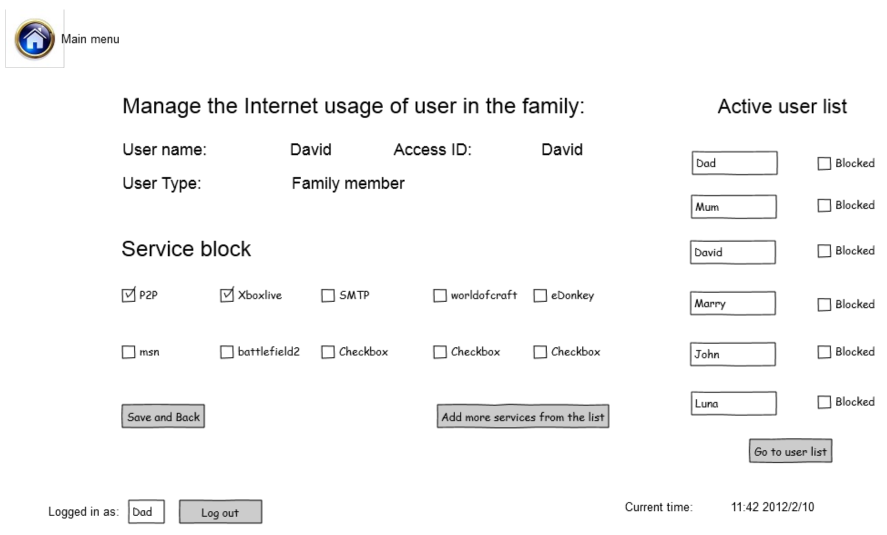
Add a new Internet user in the family

Manage family members' Internet usage

View advanced settings for Home Internet Cafe

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Manage the Internet usage of user in the family:

User name: David Access ID: David

User Type: Family member

Service block

☑ P2P ☑ Xboxlive ☐ SMTP ☐ worldofcraft ☐ eDonkey

☐ msn ☐ battlefield2 ☐ Checkbox ☐ Checkbox ☐ Checkbox

Save and Back Add more services from the list

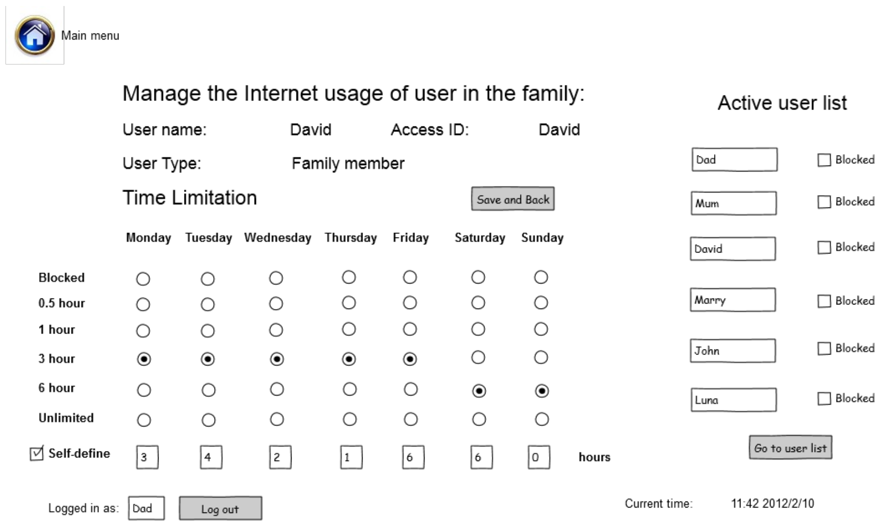
Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☐ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Go to user list

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Manage the Internet usage of user in the family:

User name: David Access ID: David

User Type: Family member

Time Limitation

Save and Back

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Blocked	○	○	○	○	○	○	○
0.5 hour	○	○	○	○	○	○	○
1 hour	○	○	○	○	○	○	○
3 hour	●	●	●	●	●	○	○
6 hour	○	○	○	○	○	●	●
Unlimited	○	○	○	○	○	○	○
☑ Self-define	3	4	2	1	6	6	0 hours

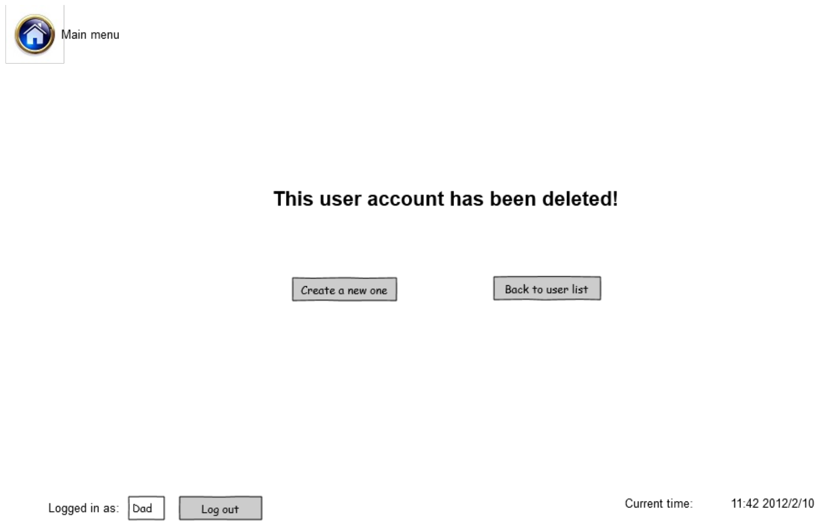
Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☐ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Go to user list

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



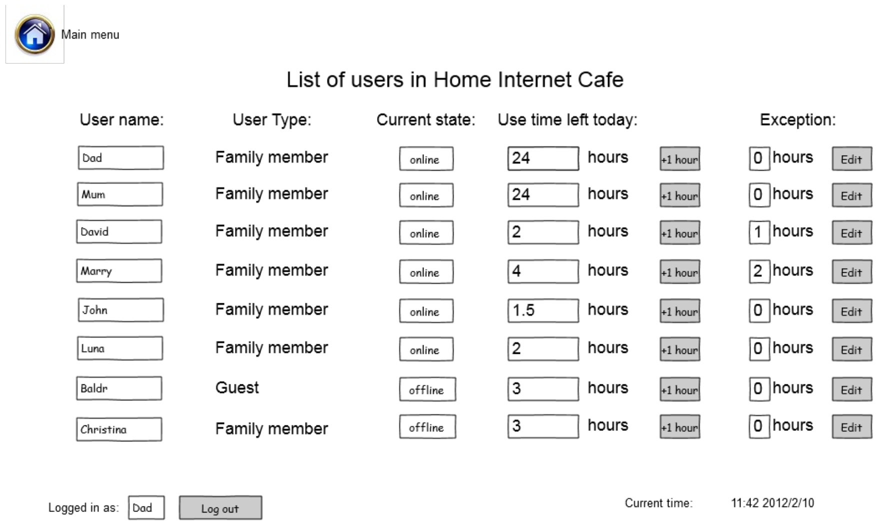
Main menu

This user account has been deleted!

Create a new one Back to user list

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



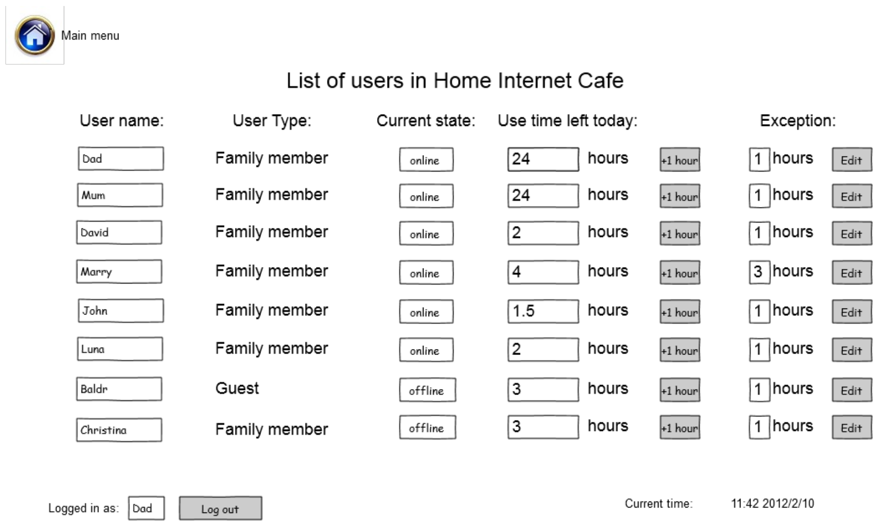
Main menu

List of users in Home Internet Cafe

User name:	User Type:	Current state:	Use time left today:			Exception:	
Dad	Family member	online	24 hours	+1 hour		0 hours	Edit
Mum	Family member	online	24 hours	+1 hour		0 hours	Edit
David	Family member	online	2 hours	+1 hour		1 hours	Edit
Marry	Family member	online	4 hours	+1 hour		2 hours	Edit
John	Family member	online	1.5 hours	+1 hour		0 hours	Edit
Luna	Family member	online	2 hours	+1 hour		0 hours	Edit
Baldr	Guest	offline	3 hours	+1 hour		0 hours	Edit
Christina	Family member	offline	3 hours	+1 hour		0 hours	Edit

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



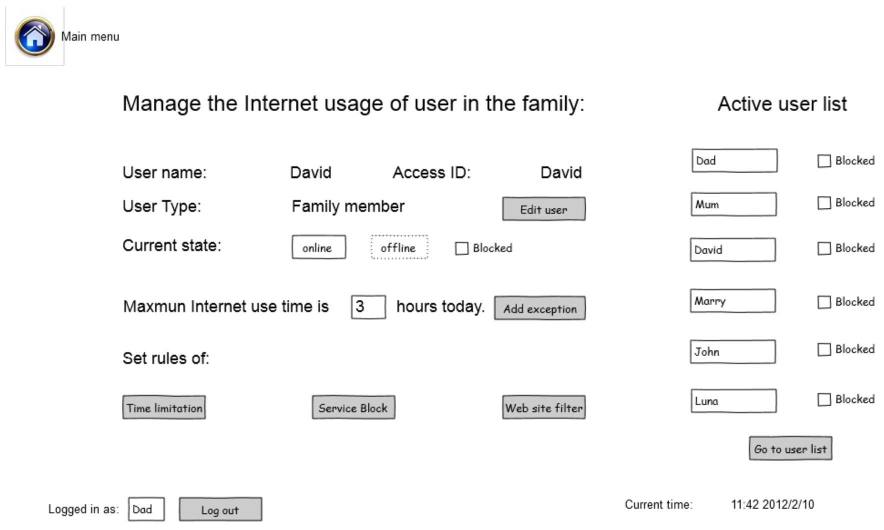
Main menu

List of users in Home Internet Cafe

User name:	User Type:	Current state:	Use time left today:			Exception:	
Dad	Family member	online	24 hours	+1 hour		1 hours	Edit
Mum	Family member	online	24 hours	+1 hour		1 hours	Edit
David	Family member	online	2 hours	+1 hour		1 hours	Edit
Marry	Family member	online	4 hours	+1 hour		3 hours	Edit
John	Family member	online	1.5 hours	+1 hour		1 hours	Edit
Luna	Family member	online	2 hours	+1 hour		1 hours	Edit
Baldr	Guest	offline	3 hours	+1 hour		1 hours	Edit
Christina	Family member	offline	3 hours	+1 hour		1 hours	Edit

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Manage the Internet usage of user in the family:

User name: David Access ID: David

User Type: Family member Edit user

Current state: online offline ☐ Blocked

Maxmun Internet use time is 3 hours today. Add exception

Set rules of:

Time limitation Service Block Web site filter

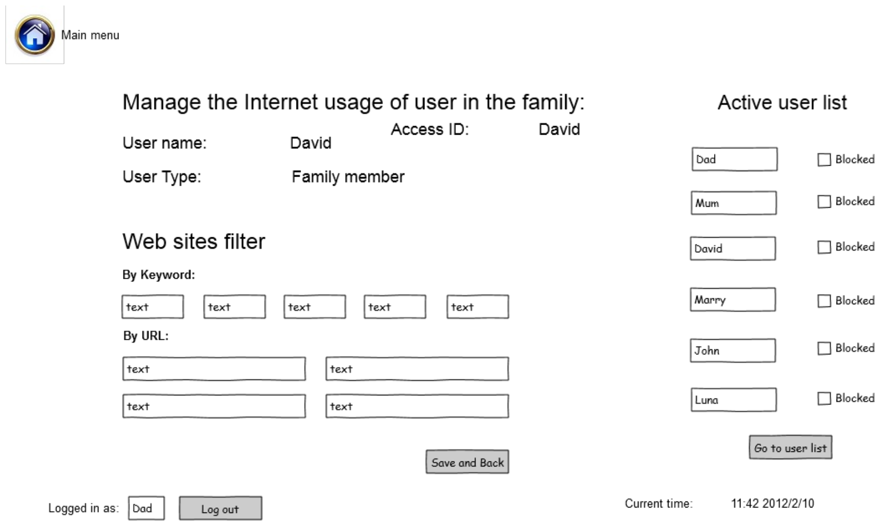
Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☐ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Go to user list

Logged in as: Dad Log out

Current time: 11:42 2012/2/10



Main menu

Manage the Internet usage of user in the family:

Access ID: David

User name: David

User Type: Family member

Web sites filter

By Keyword:

text text text text text

By URL:

text text

text text

Save and Back

Active user list

Dad	☐ Blocked
Mum	☐ Blocked
David	☐ Blocked
Marry	☐ Blocked
John	☐ Blocked
Luna	☐ Blocked

Go to user list

Logged in as: Dad Log out

Current time: 11:42 2012/2/10

Appendix D.

Paper prototype

#00

WELCOME TO PARENT CONTRL INTERNET CAFÉ

NO PARENT ACCOUNT EXISTED

CREAT A NEW ACCOUNT ?

YES #01 NO #00

"Only one account is allowed after account created. To return back to this page. Choose reset system in account control after log in as a parent account "

Creat new parent Account. #01

USER NAME	DADDY/MUM	A username should - - - -
PASSWORD		your password is too simple! /
CONFIRM PASSWORD		is OK !
		is Secure !
EMAIL ADDRESS		

"This is ~~is~~ for get password back when you forget it. Please write your own address that can be accessed as the password will be sent to the address if you click "forget password".

Confirm #02 Cancel #00

#02

New PARENT ACCOUNT CREATED

Username : X X X X X X X

password : X X X X X X X X

Email Address : ______ @ ______

[Read the Guideline] #03   [Start using now] #08

#03

Home control Internet Café Guide Lines.

① Add new family member / guest to the home Internet #04

② Control family member's Internet use #05

③ Advanced Internet settings for Internet control. To be added

⇓

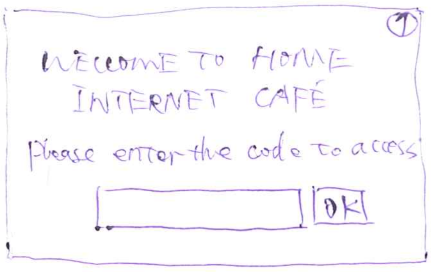
time set
password change
Remote control guideline

④ To be added.

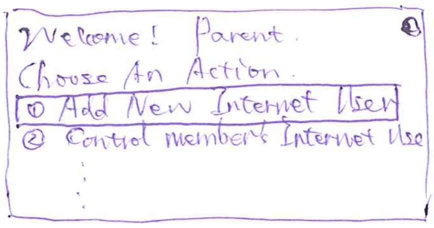
[Back to Main Page] #08   [Next] #04

#04

Add new family member/guest to home Internet.

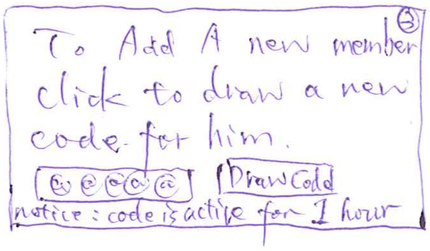

①
WELCOME TO HOME INTERNET CAFÉ
Please enter the code to access
OK

Step1: When family member/guest connected to the router and open a web browser. They will see picture ① in the screen.


②
Welcome! Parent.
Choose An Action.
① Add New Internet User
② Control member's Internet Use

Step 2: Go to log-in the parent account, you will see picture ②. Choose "add new Internet user".

Step 3: See picture ③ And choose "Draw Code For New User".


③
To Add A new member click to draw a new code for him.
@@@@@ Draw Code
notice: code is active for 1 hour

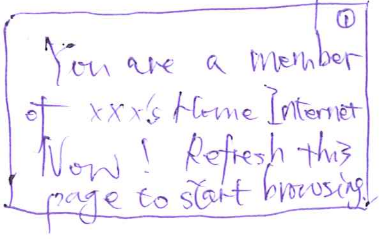
Step4: Let family member input the drawed code in picture ① to connect to the Internet.

See Configure members' Internet use then.

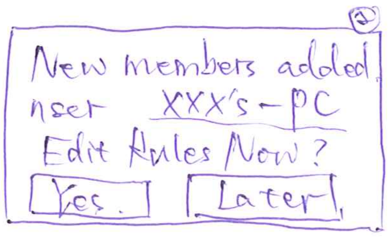
[Next Page] #05 [Back to Main] #08

Configure members' Internet Use

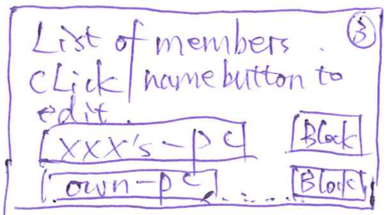
#05


①
You are a member of xxx's Home Internet Now! Refresh this page to start browsing.

After successly input the code. Family member/guest have connected to the Internet by seeing picture ①.


②
New members added. user XXX's-PC
Edit Rules Now?
Yes. Later

A picture ② will show up in parent account's screen to notice once. If you miss it, you can still go from mainpage to reach picture ③.

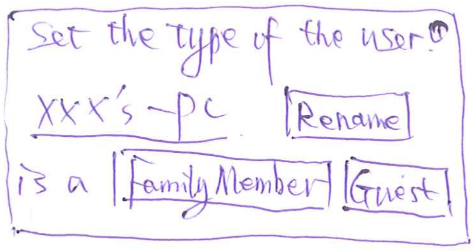

③
List of members. Click name button to edit.
XXX's-PC Block
own-PC ...... Block

Choose [Later] will return to main page.

Choose [Yes] or click name button [XXX's-PC] in picture ③ will lead you to Internet use control page.

[Next Page] #06 [Back to Main] #08

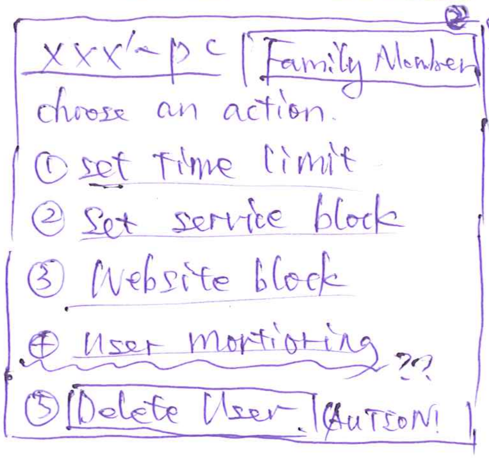
Configure members' Internet Use 2 #06.



Set the type of the user ①

XXX's-pc [Rename]

is a [Family Member] [Guest]



②

XXX'-pc [Family Member]

choose an action.

① set time limit

② Set service block

③ Website block

④ User mortioring ??

⑤ [Delete User] (AUTION!

Enter the Internet Control Page you will first see Picture ① Where you can rename the user name to help you distinguish And you can set the type: A family member will directly connected to router next time while a guest ~~router~~ should draw a new code for next connection after 24 hours (* can be changed in advanced settings).

Then we come to page ② where we can set different rules for the user.

①: set different time rules for user

②: Block different Internet Service. If you don't know. Check more Information at Wikipedia: www.wikipedia.com/.....
or leave it as default settings.

③: Block Websit by key words or URL.
For more Information check www.wikipedia.com/....

④ To be set ?

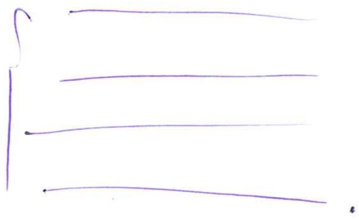
⑤ Delete the user. Think well before do it!

[Next Page] #07 | [Back to Main] #08

How to Get back to Account creat page? #07

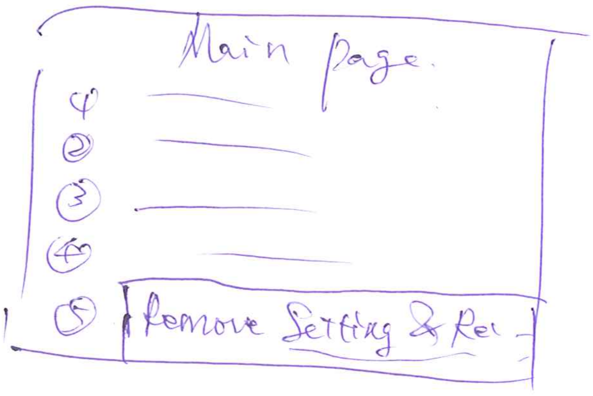
two ways:

① Reset the router. $\xrightarrow{\text{how to do it}}$



② Choose the "Remove Settings & Restart the Router To Creat new Accout." in Main Page.

Then you should Input the passwords again and confirm twice! After the Router restart. You go to the page you first see.



Main page.

④

②

③

④

⑤ [Remove Setting & Re

#08 [Back to Main].

#08

WELCOME: DADDY/MUM

Now you want to:

Add a new Internet user in the family #10

Control Family Members' Internet use. #18

Advanced Setting for Home Internet Cafe. #20

Don't know what/how to do? Read the GuideLines

WELCOME TO PARENT CONTROL INTERNET CAFE. #09.

LOG IN WITH THE ACCOUNT:

USER NAME

PASS WORD

APPLY FOR INTERNET USE:

APPLY #11

Forget PASSWORD? Click Here. To send a mail with password to pre-set email address.

Input wrong:

Wrong USR/PASSWORD.
You have ·?· times left to retry.

Input right ⇓ go to #08.

Over time: Block Access For 1 day...

#10

Add a new Internet user in the family

Add a Family Member | Add a Guest #11

Family user name: Marry

User's ID code:

MarryCooper | Set | Draw a Random

Add #12 | Delete

GuideLine: A user connected to the router will get to this page at the first time they open a browser to surf the Internet. Input the users' ID codes you set will enable the Internet access for target person in at most 10 different devices, for details, click here to read guidelines.

Welcome to Home Internet Cafe.
Input you user ID code
OK

Existing Users.

Family Members | Guest

Dad

Mum → #105

John

Marz

Add a Family Member | Add a Guest

Guest user: John's classmates "You can leave it blank"

User~~s~~'s ID code:

John's classmates | Set | Draw a Random

Add | Delete

GuideLine: A user

Welcome to Home Internet Cafe
Input you user ID code
OK

A guest's ID code will expire after 24 hours and can be used by different person in different devices.

Guest History #11

Guest | Family Members

Uncle Sam

Leo

Marz's ~~Fried~~ friends

New User added!

User: Marry

ID code: Marry_Cooper →

User Type: Family Member / Guest

Creat a new user #10

Main Menu #08

A new user has no limitation on Internet usage. Set rules and control user's access now?

~~Internet usage management~~ #18

Apply rules for this user #13

Welcome to Home Internet Cafe

Input your user ID code

⇓

open ~~target~~ the target family member's computer and open a browser. Input the ID code when you see this page to enable the Internet access.

(Connect the computer to router first).

User: John type: Family Member ID code: xx... #10 Edit

status: Active offline Blocked ✓ ☒

~~Set~~ overall rules change #14

TIME LIMIT RULES : Study For Workdays Relax at Weekend Holidays

Service Blocked : P2P #16

Website Filter : Default #17

go to devices list

Save & Back

User Status

Family Member: John.

#14

Applied overall Rules: [Add new]

Priority level

Time: Study during Workadys [Remove] [High ▽]

Relax at Weekend [Remove] [High ▽]

Holiday [Remove] [Low ▽]

High
Medium
Low

Service Blocked: [Edit] #16

p2p

Website filter [Edit] #17

default / defined

[Save & Back] [Read GuideLines]

Available Rules

#15 Study during Workdays

#15 Relax at Weekend

#15 Holiday

Click rule name to edit.

or [Creat new] #15

Time Limitation Rule ∨

#15

Name: [Study during workdays] [Rename]

[Block] [Access] user's Internet access at : ☐ Everyday

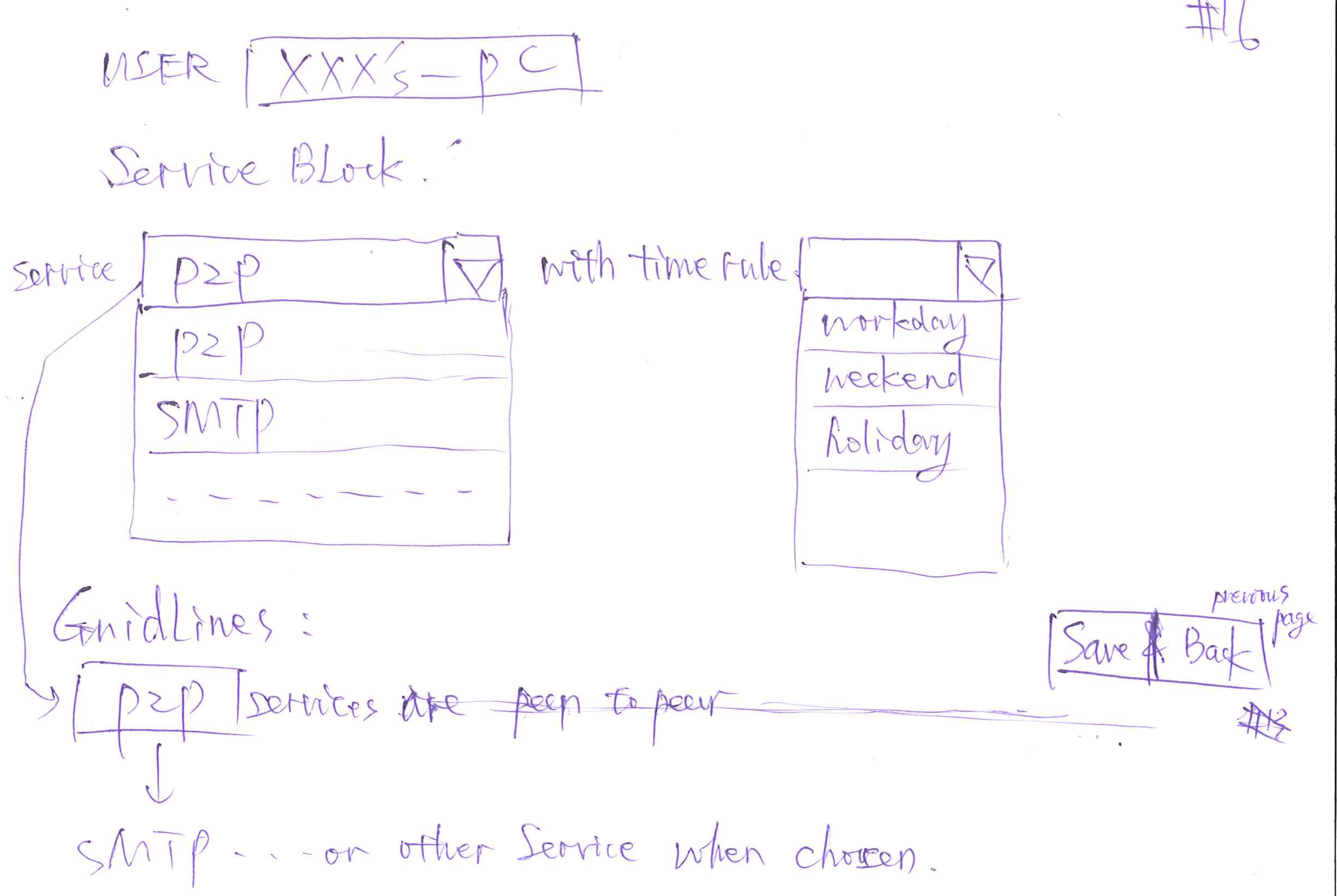
☐ Sunday ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☐ Sat

at the time: ☐ 24 hours

From 22:00 to ☑ Next Day's 15:00

Max usage time [6] half hours per day [unlimited when left blank]

[Save & Back] #14 [Delete this rule]



#16

USER XXX's—PC

Service Block.

service p2p ▽ with time rule ▽

p2p

SMTP

- - - - - - - -

workday

weekend

holiday

Guidlines:

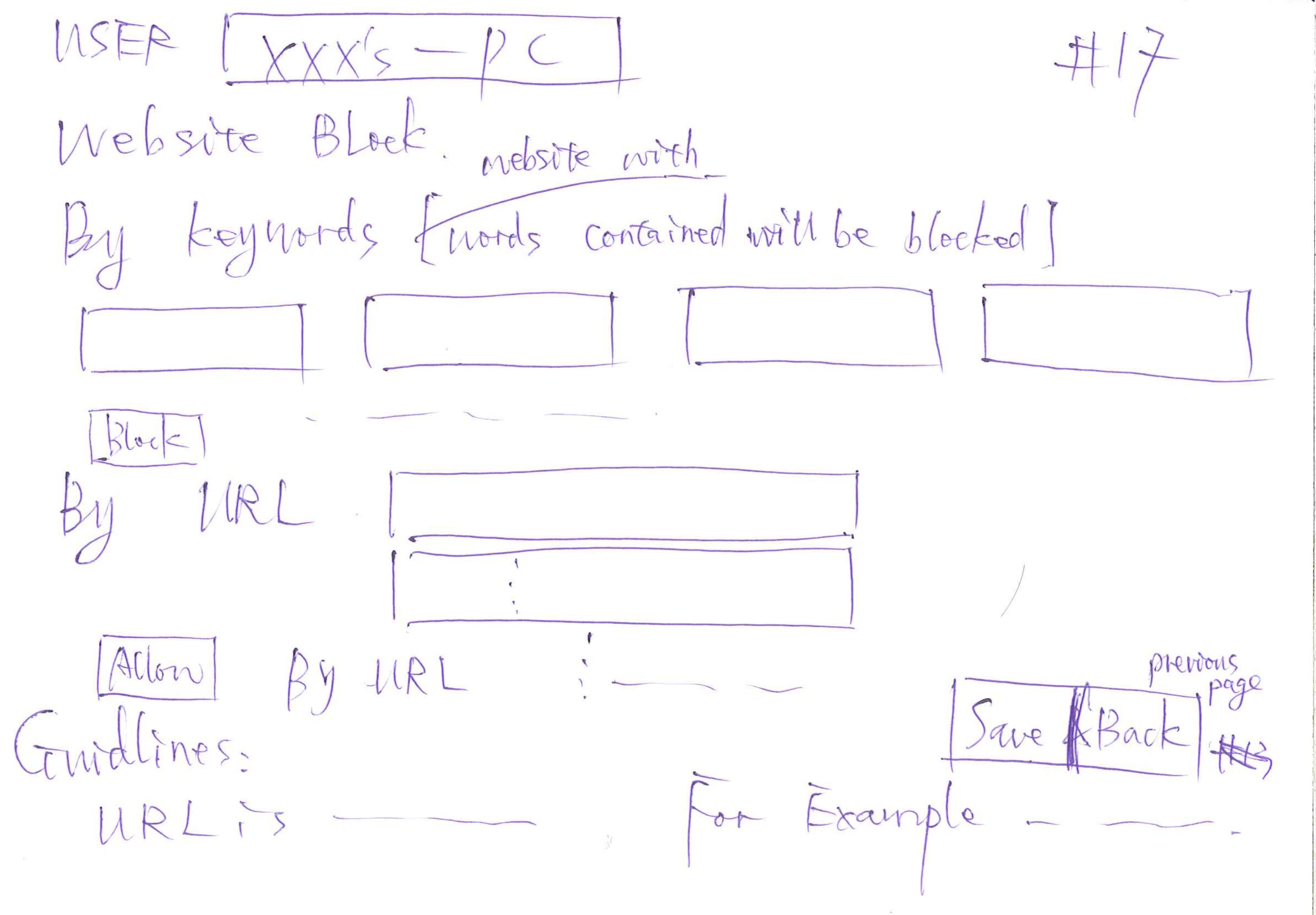
Save & Back previous page

#13

p2p services ~~are peer to peer~~ — — —

↓

SMTP . . . or other Service when chosen.



USER XXX's—PC

#17

Website Block.

By keywords [website with words contained will be blocked]

Block

By URL

Allow By URL

Guidlines:

Save & Back previous page

#13

URL is ———

For Example ———.

#18

Internet use Control

List of users [Family Member] [Guest] [All]

name:	block state:	status:	type:	online time today	#19	#13
Dad	☑	active	Parent Account	3hours	[devices list]	[configure]
Mum	☑	active	Family Member	1hour	[devices list]	[configure]
John	☑	active	Family Member	4.5hours		
Marz	☑	active	Family Member	2hours		
Marry	☑	offline	Family Member	0hour		
Naughty Boy	☒	blocked	Family Member	0hour		

#19

User: [John] type: [Family Member]

devices List:

	Mac address	IP address		
[John's Computer]	00:00:0 - - -	192.168.1.xx	[Rename]	[Block]
[John's Smartphone]				
[John's iPad]				
[xxxx...8064... Android]			[Rename]	[Block]

Tips: Don't know what device is with random number name? Block other and leave the only one acessed to find target device. Then rename it.

Advanced Settings

#20

- account management #21
- set time and date #22
- access control advanced settings #23

[Main page] [Apply]

Parent account management

#21

Current user name		[Change] →		Input password
set a new password:				
old password				
new password				
confirm new password				
current email address		[change] →		Input password

enable a new parent account ??

[Save & Back] [Cancle]

#22

Current time : H $\boxed{14}$: M $\boxed{22}$ S $\boxed{10}$ $\boxed{\text{change}}$

current date :

Feb. 2012

sun	mon	Tue	Wed			
			1	2	3	4
5	$\boxed{6}$					

$\boxed{\text{Change}}$

Time Zone : $\boxed{\text{UTC + 01:00 Olso . Berlin. Rome}}$ $\boxed{\text{Change}}$

~~Summer~~ Daylight saving time remind [✓]

$\boxed{\text{Save \& Back}}$ $\boxed{\text{Cancel}}$

Access control advanced settings: #23

- Expiring time for guest user $\boxed{24}$ hours
- Rules ~~conflict handler~~:

  when rules of ~~time limitation,~~ conflict with same periority : [ ] always choose limitation

  [✓] always choose permission
- give out notification about drop Internet acess to user when max usage time reach at $\boxed{15}$ minutes before and 1 minutes before.

$\boxed{\text{Save \& Back}}$